論文98-35S-6-4

분산 ATM 교환 제어시스템에서 프로세서간 통신 정합부에 대한 성능 분석

(Performance Analysis for the IPC Interface Part in a Distributed ATM Switching Control System)

呂 煥 根 *, 宋 光 錫 *, 盧 承 煥 **, 奇 長 根 ***

(Hwan-Geun Yeo, Kwang-Suk Song, Soong-Hwan Ro, and Jang-Geun Ki)

요 약

교환기 제어계의 구조는 전기통신 서비스에서 필요로 하는 다양한 호처리 기능을 제공하기 위하여 많은 구조적인 변화가 진행되어 왔다. 특히 분산 교환제어 환경하에서의 호처리 수행에 있어 프로세서들간의 통신에 의한 지연은 시스템의 성능에 영향을 미치는 중요한 요소 중의 하나이다. 본 논문에서는 분산 제어 구조를 갖는 ATM 교환기에서 호처리 수행에 필수적으로 요구되는 프로세서간 메시지 통신이 ATM 스위치를 통해서 이루어지는 경우, 각 프로세서내의 한 기능으로 구현되는 IPC(Inter Processor Communication) 정합부에 대한 성능 분석 모델을 제안하고, 시뮬레이션을 통해서 프로세서의 성능에 미치는 병목 요인에 대해서 검토하였다. 결과적으로, 프로세서간 통신 메시지의 입력율 변화에 따라 이를 처리하는 각 성분(자원)의 이용율과, 메시지 입력율의 변화에 따른 각 성분에서의 큐길이 및 처리 지연시간과의 관계로 부터 IPC에 관련되는 주요 성분 중 로컬 CPU가 프로세서 시스템의 최대 성능을 제한하는 주 요인이 됨을 정량적으로 확인하였다. 또한 로컬 CPU의 성능 변화에 따른 IPC 메시지 처리 지연효과와, 평균 메시지 길이의 가변에 따른 로컬 CPU의 처리 능력을 정량적으로 제시하였으며, 이 결과는 향후 프로세서의 성능 개선이나 시스템 확장을 위한 기초 자료로 활용될 수 있을 것이다.

Abstract

The control system architecture in switching systems have undergone numerous changes to provide various call processing capability needed in telecommunication services. During call processing in a distributed switching control environment, the delay effect due to communication among main processors or peripheral controllers is one of the limiting factors which affect the system performance.

In this paper, we propose a performance model for an IPC(Inter Processor Communication) interface hardware block which is required on the ATM cell-based message processing in a distributed ATM exchange system, and analyze the primary causes which affect the processor performance through the simulation. Consequently, It can be shown that the local CPU of the several components(resouces) related to the IPC scheme is a bottleneck factor in achieving the maximum system performance from the simulation results, such as the utilization of each processing component according to the change of the input message rate, and the queue length and processing delay according to input message rate. And we also give some useful results such as the maximum message processing capacity according to the change of the performance of local CPU, and the local CPU maximum throughput according to the change of average message length, which is applicable as a reference data for the improvement or expansion of the ATM control system.

* 正會員, 韓國電子通信研究院 制御시스템 研究室
(Processor Development Section, ETRI)
** 正會員, 公州大學校 情報通信工學科
(Dept. of Information & Communication Eng., Kongju National University)
*** 正會員, 公州大學校 電子工學科
(Dept. of elec. Eng., Kongju Nat'l University)
接受日字: 1997年5月23日, 수정완료일: 1998年4月10日

Ⅰ. 서 론

교환기 제어 시스템 구조는 현재와 미래의 전기통신 서비스에 의해 요구되는 다양한 호 처리 능력을 제공하기 위해 많은 변화가 이루어져 왔다. 즉, 과거에는 대부분의 제어 기능들이 주로 하나의 프로세서에서 처리되는 중앙 집중형 제어 구조를 가졌으나 최근에는 복수개의 프로세서에서 요구되는 기능들을 분산 처리하는 구조로 변화되고 있다[1-5]. 이러한 기능의 분산은 모듈단위로 독립적으로 제어가 이루어지므로 소규모에서 대규모로 시스템을 확장하는데 있어 매우 경제적인 장점을 제공한다. 반면, 각기 분산된 제어기능을 수행하기 위해서 프로세서간 메시지 상호 교환에 따른 프로세서간 통신 지연이 유발되는데 이것은 시스템 성능 저하를 초래하지 않는 범위내에서 이루어져야 한다. 이러한 프로세서간 통신방법에 대해서는 분산형 컴퓨터 통신방식에서 제안된 여러 가지 기법들이 응용되거나, 교환기의 스위치망을 이용하는 등 여러 형태로 연구되어 왔다.

특히 ATM(Asynchronous Transfer Mode) 망의 핵심 노드로서 현재 연구중인 대부분의 ATM 교환시스템의 제어계는 분산구조를 가지며, 호처리 기능을 수행할 때 분산되어 있는 프로세서간의 통신은 높은 안정성과 고속 경로를 제공해 주는 ATM 스위치망을 이용하는 추세이다.[7] 이것은 기존의 교환기 제어계에서 구성된 별도의 프로세서간 통신망을 대체하고, 통신 대역폭의 제약을 백 Mbps 이상의 물리적인 대역폭을 제공함으로써 기존 컴퓨터 통신망을 이용할 때 야기될 수 있는 물리적인 지연 요소를 거의 해결할 수 있기 때문이다. 따라서 프로세서간 통신의 지연 요소로는 송수신 되는 메시지 프로토콜의 처리 스킴이나, 메시지와 셀간의 변환능력에 의해 크게 좌우될 수 있다. 즉, ATM 교환기의 성능은 제어시스템의 프로세서간 통신 정합부의 구현 방법에 따라 시스템 성능에 결정적인 영향을 받게 된다.

본 논문은 분산 제어 구조를 갖는 ATM 교환기에서 호처리 수행에 있어 필수적으로 요구되는 프로세서간 메시지 통신이 ATM 스위치를 통해서 이루어지도록 각 메인 프로세서내에 구현되는 IPC(Inter Processor Communication) 정합부에 대한 성능 분석 모델을 제안하고, 시뮬레이션을 통해서 최대 메시지 처리능력을 정량적으로 제시하는 것을 최종 목표로 설정하였다. 본 논문은 1장 서론에 이어 2장에서는 분산 ATM 교환기의 제어계 구조와 동작에 관하여 설명하고, 이어서 3장에서는 IPC 정합부의 성능분석 모델을 제시하고 관련 입력 파라미터들을 정의한다. 4장에서 SLAM Ⅱ를 이용하여 얻어진 시뮬레이션 결과들을 검토 분석하여 최대 메시지 처리능력을 도출한다. 끝으로 5장에서는 결론 및 향후 연구과제에 대해서 기술한다.

Ⅱ. ATM 교환기의 분산 제어 구조 및 동작

그림 1은 분산 제어 구조를 갖는 ATM 교환기의 한 예에 대한 블록 구성도를 나타낸 것이다. 그림에서 보인 바와 같이 ATM 교환 시스템은 초고속망의 초기 단계에서 필요로 하는 소규모에서 망의 성숙단계에서 요구되는 대규모 서비스에 대한 적용을 고려하여 시스템 확장의 기본 단위가 되는 ALS(ATM Local Switching Subsystem)와 다수의 ALS들을 상호 연결하고 시스템 차원의 중앙 제어 기능들을 수용하는 ACS(ATM Central Switching Subsystem)로 구성되는 모듈화 구조를 갖는다. ALS는 다시 다양한 가입자 단말들을 수용할 수 있는 입출력 가입자 정합 모듈, 셀프 라우팅 기능을 제공하는 ATM 스위치 모듈과, ALS 단위로 호처리 및 관리 기능을 담당하는 CCCP(Call & Connection Control Processor) 로 크게 분류되며, ACS는 다수의 ALS들을 완전하게 상호 연결시켜 주는 다수의 ATM 스위치 모듈, 망동기 모듈, 시스템 차원의 운용 및 유지보수 기능을 담당하는 OMP(Operation & Maintenance Processor)로 각각 기능 분산되어 구성된다. 따라서 임의의 ALS내의 한 CCCP는 동일 서브시스템내 입출력 가입자 모듈의 제어기(Controller)나 다른 서브시스템에 속하는 CCCP 혹은 OMP와 IPC 경로로 제공되는 ATM 스위치를 통해 상호 메시지 교환을 수행한다. 이를 위해서 각 프로세서는 그림 1의 CCCP에 대한 세부 구성도에서 보인 바와 같이 ATM-IPC 정합부와 메인 처리부로 다시 나뉘어져 구성된다.

여기서 각 프로세서내에서 수행되는 IPC에 의한 메시지 처리 동작은 다음과 같이 세분화된 단계를 거쳐서 이루어진다. 먼저 메시지 수신절차를 살펴보면, 수신 SAR칩(SAR-R)은 ATM 스위치로부터 수신되는 ATM 셀들을 수신패킷메모리(PK-R)에 저장하기 위

해 먼저 수신제어메모리(CM-R)를 엑세스하여 필요한 정보를 얻는 절차가 필요하며, 이 정보를 이용하여 수신패킷메모리의 적절한 위치에 수신 셀의 헤더 정보를 제외한 페이로드(payload) 부분을 저장하게 된다. 만일 한 메시지에 대한 셀들의 수신이 완료되면 수신 SAR칩은 이 사실을 로컬 CPU에게 인터럽트를 통해 알리게 되며, 로컬 CPU는 인터럽트 처리 루틴에서 수신된 메시지를 수신패킷메모리로부터 이중포트메모리(DPRAM)로 옮기게 되는데 수신 SAR칩과 마찬가지로 먼저 수신제어메모리를 엑세스하여 필요한 정보를 얻은 후 수신 메시지를 옮긴다. 이중포트메모리로 옮겨진 메시지들은 메인 CPU에 의해 메인메모리(MMEM)로 옮겨지게 된다.

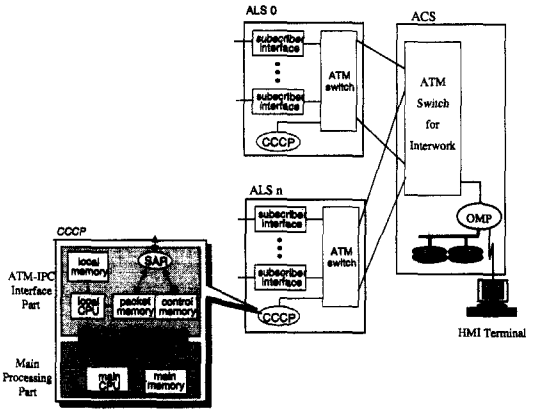
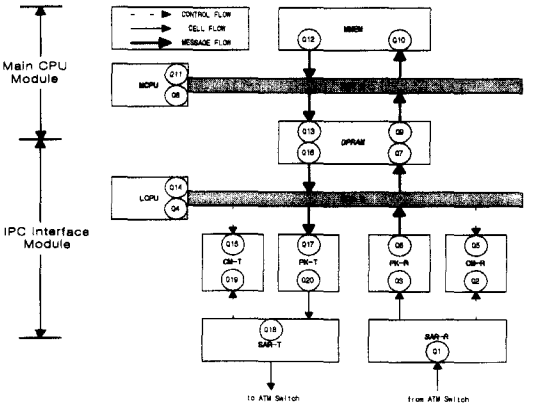


그림 1. 분산 ATM 교환시스템의 구조
Fig. 1. The architecture of a distributed ATM switching system.

메시지 송신절차의 경우, 수신 절차와 반대로 메인 CPU에 의해 메인메모리로 부터 이중포트메모리로 메시지가 옮겨지면, 로컬 CPU는 주기적인 플래그(flag) 검사 방식에 의해 이를 검출하여 이중포트메모리로부터 송신패킷메모리로 메시지를 옮기게 되고, 송신 SAR칩은 이를 IPC 링크를 통해 ATM 스위치로 전송하게 된다. 이때 수신의 경우와 마찬가지로 로컬 CPU와 송신 SAR칩은 송신패킷메모리를 엑세스하기 전에 먼저 송신제어메모리를 엑세스하는 절차가 필요하다. 이와 같은 절차에 따라 각 프로세서내에서 송수신 메시지를 처리할 때 각 구성 성분(자원)의 엑세스에 대한 우선순위가 부여된다. 즉, 패킷메모리와 제어메모리의 경우 SAR칩이 로컬 CPU에 비해 우선순위가 높게 부여되어 하나의 메시지 처리를 위해 로컬 CPU가 패킷 또는 제어메모리를 엑세스하는 도중에 SAR칩이 메모리에 대한 엑세스를 선점(preemption)할 수 있다. 메인 CPU와 로컬 CPU의 이중포트메모리에 대한 엑세스 우선순위는 같으며 트랜잭션 단위로 선입선출(FCFS) 방식이 적용된다. 또한 로컬 CPU는 평소에 이중포트메모리내의 특정 영역을 주기적으로 검사하여 메인 CPU로부터 송신메시지가 존재할 경우 이를 송신패킷메모리로 옮기는 작업을 수행하는데 만일 수신메시지가 발생하면 이 사실을 수신 SAR칩으로부터 인터럽트 형태로 통보받고 이를 처리한다. 따라서 로컬 CPU는 송신메시지와 수신메시지중 우선적으로 수신메시지를 처리한다고 볼 수 있다.

Ⅲ. IPC 성능분석을 위한 모델링 및 입력 파라미터

그림 2는 각 프로세서(CCCP)의 성능 분석을 위해 프로세서간 통신에 의한 메시지 처리에 관련되는 성분들을 중심으로 모델링한 것이다.



MCPU : Main CPU
MMEM : Main Memory
CPU : Local CPU
DPRAM : Dual Port RAM memory
CM-T/R : Tx/Rx Control Memory
PK-T/R : Tx/Rx Packet Memory
SAR : cell Segmentation And Reassembly unit

그림 2. 메인 프로세서의 IPC 메시지 처리 모델 구성도
Fig. 2. The IPC message processing model configuration of a main processor.

그림에서 사각형으로 표시된 각 CPU와 메모리들은

자원(resource)으로 모델링되고, Q 글자로 시작하는 레이블이 들어있는 작은 원들은 시뮬레이션 모델에서 수신 또는 송신 메시지가 각 자원에 대기되는 큐들을 나타낸다. 즉, 수신 메시지의 일련의 전송과정 살펴보면, Q1에서부터 Q10까지의 큐들을 차례로 거치며 처리되고, 송신 메시지의 경우 Q11에서부터 Q20까지의 큐들을 거치며 역방향으로 처리된다. 또한 각 큐들의 우선 순위는 표 1에 나타내었으며, 표 1에 나타나지 않은 큐들 사이의 우선 순위는 동일한 것으로 가정한다.

표 1. 각 자원에 대한 입출력 큐의 우선도
Table 1. Input/output queue priority of each resource.

Resources	Queue Prioritues
MCPU	Q8>Q11
LCPU	Q4>Q14
MMEM	Q10>Q12
DPRAM	Q7>Q16, Q9>Q13
CM-T	Q19>Q15
CM-R	Q2>Q5
PK-T	Q20>Q17
PK-R	Q3>Q6

IPC를 통한 메시지 처리에 관한 프로세서 모듈의 성능 평가를 위해 사용된 주요 파라미터들에 대한 정의와 각 파라미터에 적용된 가정은 다음과 같다.

* λ_{TXmsg} [msg/sec]

메인 CPU로부터 생성되는 송신메시지의 평균 발생률을 나타내며, 메시지 발생은 포아송 분포를 갖는다.

* λ_{RXcell} [msg/sec]

수신 SAR칩으로 입력되는 셀들의 평균 도착율을 나타내며, 포아송 분포를 갖는다.

* L_{msg} [cell]

로컬 CPU와 메인 CPU에서 처리되는 송수신메시지의 평균길이를 셀 수로 나타낸 것으로, 최소 1에서 최대 11 사이의 랜덤 지수분포를 가진다. 최소값과 최대값의 제한은 실제 설계된 시스템으로부터 추출한 값이다.

* L_{cell}, $L_{payload}$ [byte]

SAR칩과 ATM 스위치간에 연결된 내부 링크를 통해 송수신되는 셀의 크기는 L_{cell} = 53 바이트(라우팅 정보는 무시)로 가정하며, 헤더를 제외한 순수한 정보(payload)의 크기는 $L_{payload}$ = 48 바이트이다.

* C_{link} [bit/sec]

SAR 칩과 연결된 전송 링크의 전송율은 C_{link} = 155 Mbps로 가정한다.

* N_{okSAR} [transaction/cell]

수신 SAR칩이 수신패킷메모리에 한 셀을 저장하기 위해 수신제어메모리를 엑세스해야 하는 오버헤드 트랜잭션 수 또는 송신 SAR칩이 송신패킷메모리로부터 한 셀을 꺼내기 위해 송신제어메모리를 엑세스해야 하는 오버헤드 트랜잭션 수를 나타낸다.

* N_{okLCPU} [transaction/msg]

로컬 CPU가 하나의 송신 또는 수신 메시지를 처리하기 위해 제어메모리를 엑세스해야 하는 오버헤드 트랜잭션 수를 나타낸다.

* $T_{tranMCPU}$, $T_{tranLCPU}$, $T_{tranSAR}$ [us]

각각 메인 CPU, 로컬 CPU, 송수신 SAR칩들이 해당 버스를 사용하여 메모리를 한 번 엑세스하는(트랜잭션) 시간을 나타낸다.

* W_{busB}, W_{busC} [bytes]

IPC정합 모듈과 메인 CPU모듈내의 각 로컬 버스에서 처리되는 병렬 데이터의 대역으로 동일하게 4 바이트(32비트)로 가정한다.

Ⅳ. 성능 분석 및 결과

앞장에서 기술한 성능분석 모델을 기초로 시뮬레이션 전용언어인 SLAM II[8,9]를 이용한 network 모델 및 프로그램을 작성하여 성능평가를 수행하였다. 전체 SLAM II network 모델은 자원 모델, 입력 트래픽 발생 모델, 수신메시지 이동 모델과 송신메시지 이동 모델로 구성되었다. 자원 모델은 그림 1에 나타낸 CCCP 모듈내의 각 CPU와 메모리, SAR 칩등을 SLAM II 자원(resource) 노드로 나타낸 모델이며, 입력 트래픽 발생 모델은 수신 SAR 칩으로 입력되는 셀과 메인 CPU로부터 송신되는 메시지를 발생시키기 위한 모델이다. 수신메시지 이동 모델은 수신 SAR칩으로 입력된 셀이 메시지로 조립되고 다시 로컬 CPU를 거쳐 메인 CPU의 메인메모리로 이동하는 경로를 모델링한 것이며, 송신 메시지 이동 모델은 이와 반대되는 과정을 나타내되 주요 차이점은 송신 메시지의 경우 메인 CPU나 로컬 CPU의 점유와 반납이 트

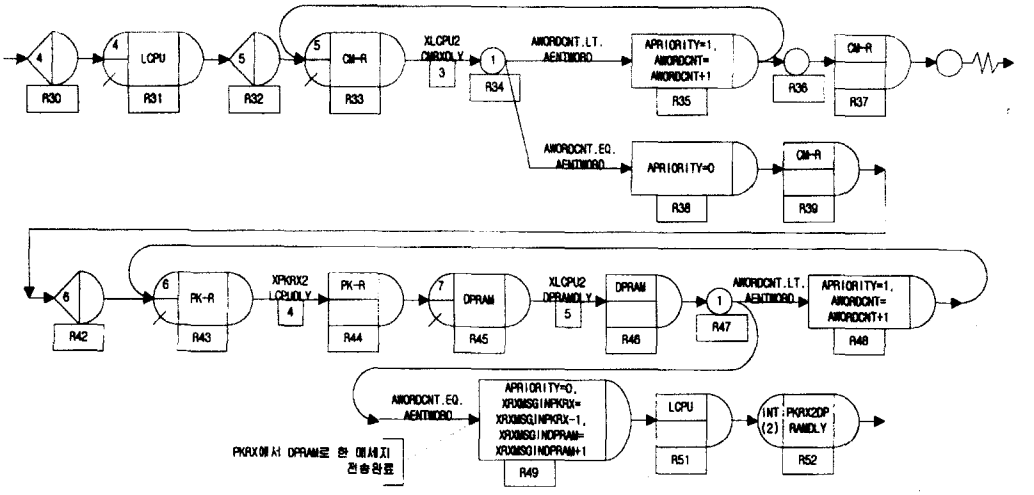


그림 3. 수신 메시지 이동 SLAM II 모델(수신패킷메모리→이중포트메모리)
Fig. 3. SLAM II model for Rx message(PK-R → DPRAM).

랜잭션 단위로 이루어지되 수신 메시지의 경우 메시지 처리시간 단위로 점유와 반납이 이루어진다는 점이다.

그림 3에 SLAM II network 모델의 일부 예로 수신메시지 이동 모델중 로컬 CPU에 의해 수신패킷메모리로부터 이중포트메모리로 수신메시지가 옮겨지는 인터럽트 처리 과정을 나타내었다. 그림에서 수신메시지는 R31 노드에서 로컬 CPU를 점유하게 되고 R32부터 R39 노드까지의 과정을 거침으로써 수신제어메모리 엑세스를 수행하며, 수신메시지 길이를 버스 크기로 나눈 횟수만큼 R43부터 R48까지의 노드들을 반복하여 거침으로써 수신패킷메모리로부터 이중포트메모리로 옮겨진 다음, R51 노드에서 로컬 CPU를 반납하게 된다. 이와 같이 한 수신메시지가 일단 로컬 CPU를 점유하게 되면 수신메시지의 처리가 완료될 때까지 로컬 CPU를 점유하게 된다. 반면에 대응되는 송신메시지 이동 모델에서는 송신메시지가 이중포트메모리로부터 송신패킷메모리로 옮겨질 때 송신메시지를 버스 크기로 나눈 횟수만큼 로컬 CPU의 점유와 반납을 반복하도록 모델링하여, 로컬 CPU가 송신메시지를 처리하는 도중에 수신메시지를 먼저 처리하는 것이 가능토록 하였으며, 이는 수신메시지의 인터럽트 처리 방식에 해당된다.

개발된 SLAM II 시뮬레이션 프로그램을 이용한 성능분석 결과로 표 2의 파라미터를 적용하여 메시지 입력율을 증가시킴에 따른 각 주요 처리 유니트의 이용율의 변화를 그림 4에 나타내었다

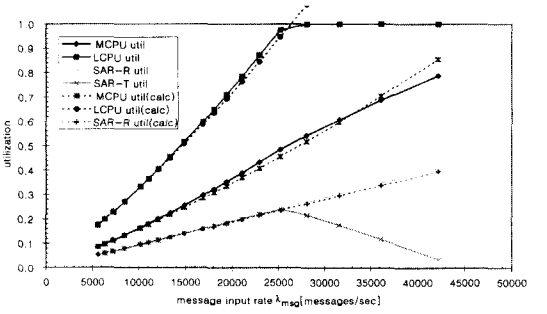


그림 4. 메시지 입력율 변화에 따른 처리 유니트의 이용율
Fig. 4. Utilization of processing units according to message input rate.

표 2. 입력 파라미터 및 적용치
Table 2. Input parameters and their values.

Input parameters	Values
$\lambda_{TXmsg} = \lambda_{RXmsg}$	가변
λ_{RXcell}	가변
L_{msg}	3.9 cells
L_{cell}	53 bytes
$L_{payload}$	48 bytes
C_{link}	155 Mbps
N_{okLCPU}, N_{okSAR}	4
$T_{tranMCPU}$	0.15 us
$T_{tranLCPU}$, $T_{tranSAR}$	0.3 us
W_{busA}, W_{busB}, W_{busC}	4 bytes

다음에는 그림 4에서 점선으로 표시된 각 처리 유니트의 점유율을 근사적으로 계산하는 방법에 대해 기술하고, 실선으로 표시된 시뮬레이션에 의한 결과와 거의 일치하고 있음을 보여준다.

먼저 그림에서 알 수 있듯이 SAR 칩은 시스템 성능의 제한 요소가 아님을 알 수 있다. 즉, SAR 칩은 그림에 나타난 범위의 입력 메시지율을 충분히 처리할 수 있는 능력이 있어 큐잉 지연이 거의 발생하지 않는다. 따라서 SAR 칩의 이용율은 아래 식들에 의해 간단히 구할 수 있다.

먼저 수신 SAR칩이 하나의 수신 셀을 처리하는 시간은 식 (1)과 같이 구해질 수 있다.

$$T_{SAR-R} = N_{ohSAR} \times T_{tranSAR} + \frac{L_{payload}}{W_{busA}} \times T_{tranSAR} \quad (1)$$

그런데 수신 SAR칩은 수신 셀을 수신패킷메모리에 저장할 때 로컬 CPU보다 제어 및 패킷메모리에 대한 엑세스 우선 순위가 높기 때문에 항상 수신 셀이 발생하면 곧바로 제어 및 패킷메모리를 엑세스 할 수 있고 또한 엑세스 중에 로컬 CPU로 부터 방해를 받지 않는다. 따라서 수신 SAR 칩의 이용율은 아래 식 (2)에 의해 구해질 수 있다.

$$\begin{aligned} U_{SAR-R} &= T_{SAR-R} \times \lambda_{RXcell} \\ &= T_{SAR-R} \times \lambda_{RXmsg} \times L_{msg} \end{aligned} \quad (2)$$

송신 SAR 칩의 이용율 또한 위의 식과 같은 방법으로 구할 수 있으며, 송신메시지와 수신메시지의 평균 도착율 및 평균 메시지 길이가 같다고 가정하였기 때문에 동일한 결과를 얻을 수 있다.

로컬 CPU의 이용율은 다음과 같이 구해질 수 있다. 로컬 CPU는 수신메시지와 송신메시지를 모두 처리하기 때문에 로컬 CPU의 이용율은 수신메시지에 의한 이용율과 송신메시지에 의한 이용율로 나누어 고려하기로 한다.

먼저 로컬 CPU가 수신제어메모리와 수신패킷메모리를 엑세스할 때 수신 SAR칩으로부터 방해를 받지 않는다고 가정할 경우 로컬 CPU가 하나의 수신메시지를 처리하는데 걸리는 시간은 식 (3)과 같이 수신제어메모리를 엑세스하는 오버헤드 시간에 수신패킷메모리와 이중포트메모리를 엑세스하는 시간을 더하여 구할 수 있으며, 식에서 2가 곱해진 것은 로컬 CPU가 수신패킷메모리로부터 메시지의 4바이트(버스 B의 크기에 해당)를 읽은 후 다시 이 4 바이트를 이중포트메모리에 쓰기 때문이다.

$$T'_{RxLCPU} = N_{ohLCPU} \times T_{tranLCPU} + \frac{L_{msg} \times L_{payload}}{W_{busB}} \times T_{tranLCPU} \times 2 \quad (3)$$

로컬 CPU와 수신 SAR칩이 수신제어/수신패킷메모리를 동시에 엑세스하는 경우는 로컬 CPU가 수신제어/수신패킷메모리를 엑세스하는 시간이 수신 SAR칩이 엑세스하는 시간보다 훨씬 길기 때문에 그림 5에 나타낸 것과 같이 구간 A나 구간 B에서 수신 SAR칩이 수신제어/수신패킷메모리에 대한 엑세스를 시작하는 경우로 나누어 생각할 수 있다. 이와 같이 동시에 엑세스를 하려고 하는 경우에는 수신 SAR칩의 엑세스 우선순위가 높기 때문에 로컬 CPU는 수신 SAR칩의 엑세스가 끝날 때까지 지연된다. 따라서 로컬 CPU가 하나의 수신메시지를 처리하는데 걸리는 시간은 평균적으로 식 (4)와 같다. 식 (4)에서 두 번째항은 구간 A에서 수신 SAR칩의 엑세스가 시작되는 경우에 로컬 CPU의 엑세스 지연시간을 나타내며, 세 번째 항은 구간 B에 대한 것이다.

$$T_{RxLCPU} = T'_{RxLCPU} + \frac{T_{SAR-R}}{1/\lambda_{RXcell}} \times \frac{T_{SAR-R}}{2} + \frac{T'_{RxLCPU}}{1/\lambda_{RXcell}} \times T_{SAR-R} \quad (4)$$

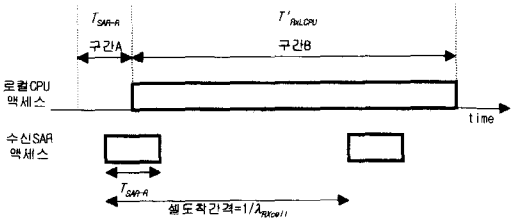


그림 5. 로컬 CPU와 수신 SAR칩이 수신제어/패킷메모리를 동시 엑세스 할 경우

Fig. 5. In case that both LCPU and SAR-R make access to the CM-R/PK-R at the same time.

구간 A는 로컬 프로세서가 수신제어/수신패킷 메모리 엑세스를 시작하고자 할 때 이미 수신 SAR가 수신제어/수신패킷 메모리를 엑세스하고 있는 중인 경우를 나타내며, 구간 B는 로컬 프로세서가 수신제어/수신패킷 메모리를 엑세스하고 있는 도중에 수신 SAR가 수신제어/수신패킷 메모리 엑세스를 시작하는 경우를 나타낸다.

로컬 CPU와 메인 CPU가 동시에 이중포트메모리를 엑세스하고자 하는 경우를 고려해야 하나, 뒤에서 메인 CPU의 이용율을 구할 때 기술한 이유로 여기서는 무시한다.

따라서 수신메시지에 의한 로컬 CPU 이용율은 식 (5)와 같이 구할 수 있다.

$$U_{RxLCPU} = T_{RxLCPU} \times \lambda_{RXmsg} \tag{5}$$

다음에는 로컬 CPU가 하나의 송신메시지를 처리하는데 소요되는 이용율을 구하면 다음과 같다.

먼저 송신 SAR칩이 로컬 CPU로부터 넘겨받은 하나의 송신 메시지를 처리하기 위해 걸리는 시간을 계산해보면 식 (6)과 같다.

$$T_{TxSAR-T} = (N_{ohSAR} \times T_{tranSAR} + \frac{L_{payload}}{W_{busA}} \times T_{tranSAR}) \times L_{msg} \tag{6}$$

수신메시지의 처리 경우와 같이 송신 SAR칩이 로컬 CPU에 비해 송신제어/송신패킷메모리에 대한 엑세스 우선순위가 높으므로, 로컬 CPU가 송신메시지를 처리하는데 걸리는 시간은 식 (7)과 같이 계산할 수 있다. 단, 로컬 CPU가 송신메시지를 처리하는 동안 수신메시지를 처리하기 위해 걸리는 인터럽트는 여기서 고려하지 않으며, 그 이유는 인터럽트 처리시간, 즉 수신메시지 처리시간은 위에서 수신메시지에 의한 이용율 계산에 반영되었기 때문이다. 또한 인터럽트에 따른 오버헤드도 없다고 가정하였다.

$$T_{TxLCPU} = T'_{TxLCPU} + \frac{T_{TxSAR-T}}{1/\lambda_{TXmsg}} \times \frac{T_{TxSAR-T}}{2} + \frac{T'_{TxLCPU}}{1/\lambda_{TXmsg}} \times T_{TxSAR-T} \tag{7}$$

따라서 송신메시지에 의한 로컬 CPU 이용율은 식 (8)과 같이 구할 수 있다.

$$U_{TxLCPU} = T_{TxLCPU} \times \lambda_{TXmsg} \tag{8}$$

최종적으로 로컬 CPU의 이용율은 식 (9)에 의해 구할 수 있다.

$$U_{LCPU} = U_{RxLCPU} + U_{TxLCPU} \tag{9}$$

다음에는 메인 CPU의 이용율을 구한다.

메인 CPU가 메시지를 처리할 때 로컬 CPU가 이중포트메모리를 동시에 엑세스하는 경우는 메인 CPU의 이중포트메모리 엑세스 시간이 로컬 CPU의 엑세스 시간보다 짧기 때문에 그림 6에 나타낸 것과 같이 로컬 CPU의 이중포트메모리 엑세스 시작 시점에 따라 3가지 경우로 나누어 생각할 수 있다. 구간 A에서 로컬 CPU의 엑세스가 시작되면 메인 CPU 엑세스의 앞 일부와 로컬 CPU 엑세스의 뒤 일부가 겹치게 되고, 구간 B에서 시작되면 메인 CPU 엑세스 전구간이 겹치며, 구간 C에서 시작되면 메인 CPU의 뒤 일부와 로컬 CPU의 앞 일부분이 겹치게 된다.

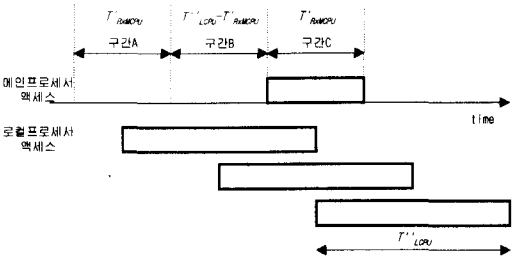


그림 6. 메인 CPU와 로컬 CPU가 이중포트메모리를 동시에 엑세스 할 경우

Fig. 6. In case that both MCPU and LCPU make access to the DPRAM at the same time.

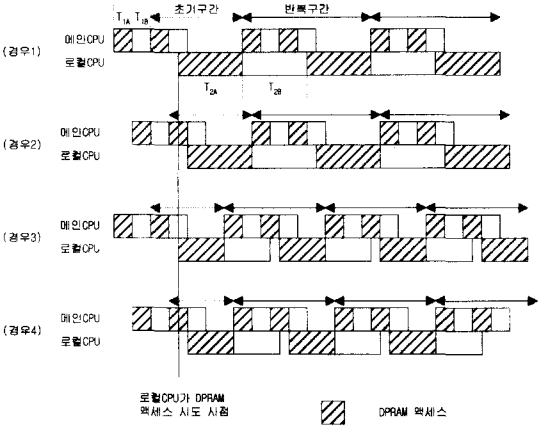


그림 7. 메인 CPU와 로컬 CPU가 이중포트메모리를 동시에 엑세스 할 때의 지연

Fig. 7. The delay due to the same access of both MCPU and LCPU to DPRAM.

동시 엑세스가 발생하는 경우의 엑세스 지연에 대해 보다 상세히 고려해보면 그림 7에 나타낸 것과 같이 T_{1A}, T_{1B}, T_{2A}, T_{2B}의 크기에 따라 4가지 경우로 구분할 수 있다. 그림에서 T_{1A}는 메인 CPU의 이중포트메모리 엑세스를, T_{1B}는 메인 CPU의 메인메모리 엑세스를, T_{2A}는 로컬 CPU의 이중포트메모리 엑세스를, T_{2B}는 수신패킷메모리 엑세스를 각각 나타낸다. 그림에서 알 수 있듯이 메인 CPU와 로컬프로세서의 이

중포트메모리는 선입선출 방식에 의해 서비스됨을 알 수 있고, 일단 동시 엑세스가 발생하면 초 구간에서 지연이 발생할 수도 있고 반복구간에서 지연이 발생할 수도 있는데, 초기구간은 반복구간에 비해 무시할 수 있다.

그런데 일반적으로 메인 CPU의 메모리 엑세스 시간은 로컬 CPU의 메모리 엑세스 시간보다 적어도 2배 이상은 빠르며, 따라서 T_{1A}, T_{1B}는 T_{2A}, T_{2B}에 비해 상대적으로 매우 작다. 따라서 그림에서 확인할 수 있듯이 반복구간에서 로컬 CPU의 이중포트메모리에 대한 엑세스 지연시간은 경우 1과 경우 2에서는 지연이 없으며, 경우 3과 4에서는 지연이 약간 있으나 메인 CPU의 지연에 비해서 상대적으로 작다. 앞에서 로컬 CPU의 이용율을 구할 때 이중포트메모리에 대한 동시 엑세스의 경우를 무시한 것은 주어진 파라미터 값이 최소한 $2\times T_{1A}=2\times T_{1B}=T_{2A}\leq T_{2B}$ 이므로 경우 3, 4에 속한다 하더라도 지연시간을 무시할 수 있기 때문이다.

반대로 메인메모리의 경우에는 로컬 CPU가 이중포트메모리를 엑세스하지 않는 T_{2B} 시간동안 이중포트메모리를 엑세스하다가 T_{2A} 시간동안 지연되는 과정을 반복함으로 상대적으로 지연이 매우 크다. 따라서 이와 같은 지연을 고려하여 메인 CPU의 메시지 처리시간을 구하되, 로컬 CPU에서와 마찬가지로 수신 메시지와 송신 메시지 처리의 경우를 나누어 고려하기로 한다.

먼저 로컬 CPU가 하나의 수신 메시지를 처리하기 위해 걸리는 평균시간은 앞에서 로컬 CPU의 점유율을 구할 때 사용했던 식 (4)에 나타낸 것과 같이 T_{RxLCPU}이므로, 로컬 CPU가 하나의 수신 메시지를 처리하기 위해 이중포트메모리를 엑세스하는 평균 시간은 식 (10)과 같이 계산할 수 있다.

$$T''_{LCPU} = T_{RxLCPU} - N_{ohLCPU}\times T_{tranLCPU} \tag{10}$$

또한 로컬 CPU가 하나의 수신 메시지를 처리하기 위해 이중포트메모리를 엑세스하는 회수는 식 (11)과 같다.

$$N_{LCPU} = \frac{L_{msg}\times L_{payload}}{W_{busB}} \tag{11}$$

따라서 그림 7에서 T_{2A}와 T_{2B}는 식 (12)와 같이 구할 수 있다.

$$T_{2A} = T_{tranLCPU},\quad T_{2B} = \frac{T''_{LCPU}}{N_{LCPU}} - T_{2A} \tag{12}$$

메인 CPU는 로컬 CPU의 T_{2B} 시간동안 이중포트메모리를 엑세스할 수 있으므로 이 T_{2B} 시간동안 메인 CPU가 이중포트메모리를 엑세스하는 평균 횟수를 구해보면 식 (13)과 같다. 식 (13)의 결과가 의미하는 것은 그림 6에서 메인 CPU와 로컬 CPU의 메시지 처리시간이 겹치는 구간에서는 개략적으로 메인 CPU가 T_{2B} 시간동안 $N_{MCPUinT_{2B}}$ 횟수만큼 이중포트메모리를 엑세스하고 난 후 T_{2A} 시간만큼 지연되었다가 다시 T_{2B} 시간동안 이중포트메모리를 엑세스하는 과정을 반복한다는 것이다.

$$N_{MCPUinT_{2B}} = \frac{T_{2B}}{T_{1A}+T_{1B}} \tag{13}$$

따라서 위의 그림 6의 구간 C에 대해 메인 CPU의 엑세스가 지연되는 시간은 식 (14), 식 (15)와 같이 계산할 수 있다.

$$N_{MCPU} = \frac{L_{msg}\times L_{payload}}{W_{busC}} \tag{14}$$

$$D_{MCPUinC} = \frac{N_{MCPU}/2}{N_{MCPUinT_{2B}}}\times T_{2A} \tag{15}$$

A구간에 대한 메인 CPU의 지연시간은 C구간과 마찬가지로 생각할 수 있으며, B구간에 대해서는 메인 CPU와 로컬 CPU의 이중포트메모리 엑세스 구간이 완전히 겹치므로 지연시간을 구하는 식 (15)에서 $N_{MCPU}/2$대신 N_{MCPU}를 대입하여 식 (16)과 같이 구할 수 있다.

$$D_{MCPUinB} = \frac{N_{MCPU}}{N_{MCPUinT_{2B}}}\times T_{2A} \tag{16}$$

또한 구간 A, B, C에서 로컬메모리의 이중포트메모리 엑세스가 시작될 확율은 식 (17), 식 (18)과 같이 계산할 수 있다.

$$P_{inA} = P_{inC} = \frac{T'_{RxMCPU}}{1/\lambda_{RXmsg}} \tag{17}$$

$$P_{inB} = \frac{T''_{LCPU} - T'_{RxMCPU}}{1/\lambda_{RXmsg}} \tag{18}$$

단, $$T'_{RxMCPU} = \frac{L_{msg}\times L_{payload}}{W_{busC}}\times T_{tranMCPU}\times 2 \tag{19}$$

따라서 메인 CPU가 한 수신메시지를 처리할 때 기대되는 평균 지연시간은 식 (20)과 같고, 수신메시지에 의한 메인 CPU의 이용율은 식 (21)과 같다.

$$D_{MCPU} = P_{inA} \times D_{MCPUinA} + P_{inB} \times D_{MCPUinB} + P_{inC} \times D_{MCPUinC} \quad (20)$$

$$U_{RxMCPU} = (T'_{RxMCPU} + D_{MCPU}) \times \lambda_{RXmsg} \quad (21)$$

송신메시지에 의한 메인 CPU의 이용율은 수신메시지에 의한 이용율을 구할 때 사용한 식 (10)부터 식 (21)까지에 송신메시지에 대한 파라미터 값들을 대입하면 얻을 수 있으며, 따라서 전체 송수신 메시지에 의한 메인 CPU의 이용율은 식 (22)와 같이 구할 수 있다.

$$U_{MCPU} = U_{RxMCPU} + U_{TxMCPU} \quad (22)$$

이상과 같은 식들에 의해 구한 각 처리 성분의 이용율과 시뮬레이션 결과로 얻은 이용율은 그림 4에서 확인할 수 있듯이 거의 일치함을 확인할 수 있다. 단지, 메시지 입력율이 초당 25,000개 이상일 경우 송신 SAR 칩의 이용율이 급격히 감소하여 계산치와 오차가 있는 것은 뒤에 밝혀지겠지만 로컬 CPU에서 병목현상이 발생하고 로컬 CPU는 수신메시지를 송신메시지에 우선하여 처리하므로, 메시지 입력율을 계속 증가할 경우 송신 SAR칩으로 로컬 CPU가 송신 메시지를 넘겨주지 못하여 계산식에서 가정한 메시지 도착율보다 훨씬 작아지기 때문이다.

그림 8에는 전체 시스템 성능에 영향을 미치는 요소를 도출하기 위해 메시지 입력율 증가에 따른 각 큐들의 길이를 나타내었다. 그림에서 확인할 수 있듯이 메시지 입력율이 약 23000개/초에 근접할 경우, 송신메시지가 로컬 CPU의 서비스를 기다리는 큐의 길이가 급격하게 증가함을 알 수 있으며, 따라서 이 지점에서 프로세서 시스템의 성능 병목요소가 로컬 CPU임을 알 수 있다. 또한 수신메시지에 비해 송신메시지가 로컬 CPU의 처리를 기다리는 큐의 길이가 갑자기 증가하는 이유는 로컬 CPU가 송신메시지를 처리하는 도중에 수신메시지가 입력되면 우선적으로 인터럽트 처리를 수행하게 된다. 이것은 송신메시지 처리보다 수신메시지의 처리 우선순위가 높기 때문이다.

이와 같은 사실은 그림 9에서 메시지 입력율 변화에 대한 로컬 CPU의 메시지 처리에 대한 결과에서도 확인할 수 있다. 그림 9에서 점선으로 표시된 min delay는 전혀 지연이 없다고 할 때 한 메시지를 처리하는데 걸리는 최소시간을 뜻한다.

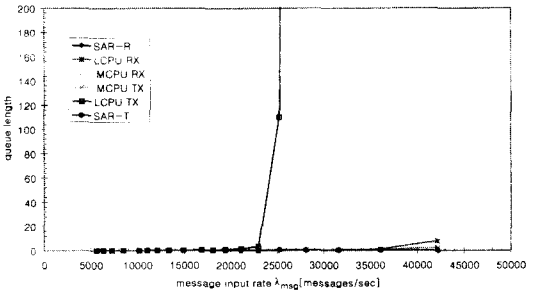


그림 8. 메시지 입력에 따른 각 자원의 큐 길이
Fig. 8. Queue length of each resource vs. message input rate.

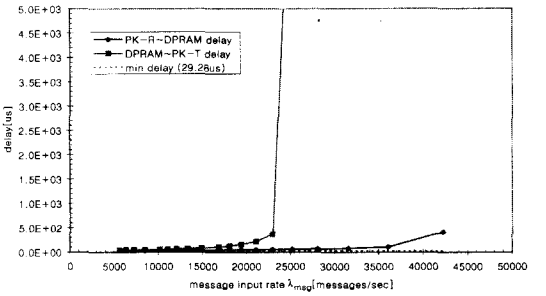


그림 9. 메시지 입력율에 대한 로컬 CPU의 메시지 처리 지연
Fig. 9. Message processing delay of Local CPU according message input rate.

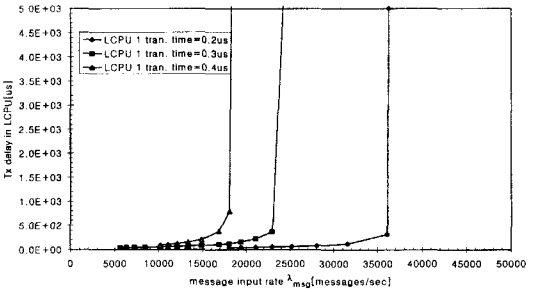


그림 10. 로컬 CPU의 성능변화에 따른 송신메시지 처리 지연시간 변화
Fig. 10. Tx message processing delay of LCPU due to the change of LCPU performance.

그림 10은 로컬 CPU 성능($T_{tranLCPU}$) 변화에 따른 송신메시지 처리 지연시간 변화 결과를 나타내었다. 그림 10에서 알 수 있듯이 로컬 CPU의 한 트랜잭션 시간이 0.3us에서 0.2us로 개선될 경우 최대 메시지 처리 용량은 평균 약 23000메시지/초에서 36000메시

지/초로 약 1.57배로 증가됨을 알 수 있다. 반대로 로컬 CPU의 한 트랜잭션 시간이 0.4us가 될 경우 최대 메시지 처리용량이 약 17000메시지/초로 감소되는 현상을 예상할 수 있다.

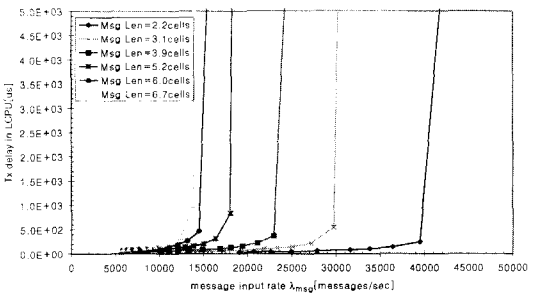


그림 11. 평균 메시지 길이에 따른 로컬 CPU의 송신 메시지 처리 지연시간

Fig. 11. Tx. message processing delay of LCPU according to average message length.

그림 11에는 메시지의 평균 길이가 변화할 경우에 로컬 CPU의 송신 메시지 처리 지연시간 변화를 나타내었다. 그림에서 메시지의 평균 길이가 길어질수록 메시지 입력율이 작은 지점으로 이동해서 로컬 CPU의 송신메시지 처리 지연시간이 급격하게 증가됨을 볼 수 있다. 이 그림으로부터 각 메시지 평균 길이에 대한 로컬 CPU의 최대 메시지 처리 용량을 구한 결과는 그림 12에서 보인 바와 같이 메시지 평균길이와 로컬 CPU의 처리성능관계가 반비례함을 알 수 있다.

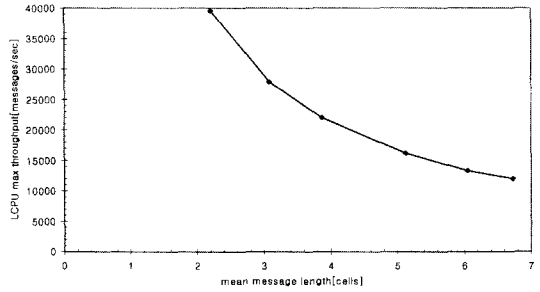


그림 12. 메시지의 평균길이에 따른 로컬 CPU의 최대 처리용량

Fig. 12. LCPU maximum processing capacity vs. mean message length.

V. 결 론

본 논문은 ATM 교환시스템의 분산 제어계 구조에서 호처리 기능을 담당하는 메인 프로세서(CCCP)내의 IPC 정합부에 대해 메시지 처리에 관계되는 구성 성분을 모델링하여 시뮬레이션과 유도식에 의한 해석적인 방법으로 성능분석을 수행하였다. 시스템 성능의 병목 요소를 밝혀내기 위해 메시지 입력율 변화에 따른 처리 성분들의 이용율을 구하였다. 또한 메시지 입력율 변화에 따른 각 처리 성분의 큐 길이 변화 및 송수신 메시지 처리 지연시간 변화 관계를 살펴 보았다. 결과적으로 IPC 메시지 처리에 관련되는 구성 성분들 가운데 로컬 CPU가 시스템 성능의 병목 요소가 되었으며, 본 시뮬레이션 결과 병목 현상이 일어나기 바로 전에서의 최대 메시지 처리용량은 약 23000 메시지/초로 나타났다. 특히 로컬 CPU의 송수신 메시지 처리 스킴에 있어서 송신메시지 처리시에 병목현상이 보다 심각하게 발생하였다. 이 외에도 각 CPU의 성능 변화와 IPC 메시지 평균길이의 변화에 따른 프로세서 모듈의 최대 메시지 처리능력을 제시함으로써 향후 ATM 교환기의 서비스 처리능력에 따른 메인 프로세서에서의 IPC 메시지 처리와 관련한 성능 개선시 기초적인 참조 자료로 활용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] E. Dutkiewicz, G. Anido, "Performance Evaluation of Hierarchical Control Systems in Switching Exchanges," *A.T.R.* Vol. 28, No. 1, pp. 15-24, 1994.

[2] E. Dutkiewicz, G. Anido, "Optimisation of Distributed Exchange Control Systems," *A.T.R.* Vol. 29, No. 2, pp. 1-13, 1995.

[3] Sakurai, Y., et al., "ATM Switching System for B-ISDN," *Hitachi Review*, Vol. 40, pp. 193-198, 1991.

[4] Suzuki, K., et al, "An ATM Switching System - Development and Evaluation," *NEC Research & Development*, Vol. 32, pp. 242-251, April, 1991.

[5] Thomas M. Chen, Stephen S. Liu, "ATM Switching Systems," Artech House, Inc., 1995.

[6] Shuichi Sumita, "Performance Analysis of Interprocessor Communications in an Electronic Switching System with

Distributed Control", *Performance Evaluation 9* (1988/89), pp. 83-91, 1989.

[7] Yong Boo Kim, Soon Seok Lee, Chang Hwan Oh, Young Sun Kim, Chimoon Han, Chu Hwan Yim, "An Architecture of Scalable ATM Switching System and Its Call Processing Capacity Estimation", *ETRI Journal*, Vol. 18, No. 3, pp. 107-125, Oct., 1996.

[8] A. Alan B. Pritsker, "Introduction to Simulation and SLAM II," John Wiley & Sons, 1995.

[9] A. Alan B. Pristsker, C. Elliott Sigal, R. D. Jack Hammesfahr, "SLAM II Network Models for Decision Support," Prentice-Hall, Inc., 1989.

저 자 소 개

呂 煥 根(正會員)

1957년 10월 20일생. 1981년 2월 : 경북대학교 전자공학과 졸업(공학사). 1983년 2월 : 경북대학교 대학원 전자공학과(공학석사). 1992년 3월 ~ 현재 : 경북대학교 대학원 컴퓨터공학과(박사과정). 1993년 3월 ~ 1994년 2월 : Univ. of Maryland SRI 객원연구원. 1983년 3월 ~ 현재 : 한국전자통신연구원 제어시스템연구실 책임연구원. ※ 주관심분야 : Fault Tolerant Computing System, Distributed Computer Architecture, ATM Switching System

宋 光 錫(正會員)

1953년 10월 23일생. 1979년 2월 : 고려대학교 전자공학과 졸업(공학사). 1981년 8월 : 고려대학교 대학원 전자공학과(공학석사). 1992년 2월 : 고려대학교 대학원 전자공학과(공학박사). 1992년 1월 ~ 1992년 12월 : Georgia Institute of Technology 객원연구원. 1982년 7월 ~ 현재 : 한국전자통신연구원 제어시스템연구실장. ※ 주관심분야 : Fault Tolerant Computing System, Distributed Computer Architecture, ATM Switching System

盧 承 燁(正會員)

1962년 8월 19일생. 1987년 8월 : 고려대학교 전자공학과 졸업(공학사). 1989년 8월 : 고려대학교 대학원 전자공학과(공학석사). 1993년 8월 : 고려대학교 대학원 전자공학과(공학박사). 1994년 3월 ~ 현재 : 공주대학교 정보통신공학과 조교수. ※ 주관심분야 : 컴퓨터 구조, ATM 교환시스템

奇 長 根(正會員)

1961년 7월 15일생. 1986년 2월 : 고려대학교 전자공학과 졸업(공학사). 1988년 2월 : 고려대학교 대학원 전자공학과(공학석사). 1992년 2월 : 고려대학교 대학원 전자공학과(공학박사). 1992년 3월 ~ 현재 : 공주대학교 전자공학과 부교수. ※ 주관심분야 : 컴퓨터 통신 및 프로토콜, ATM 교환시스템