[6-13] TDX-10 자국호처리 소프트웨어의 성능평가 및 분석

°한인탁* 강창언**
*한국전자통신연구소, **연세대학교 전자공학과

Performance Evaluation and Analysis of the Internal Call Processing Software in the TDX-10 Electronic Switching System

°In-Tark Han*, Chang-Eon Kang**
*Electronics and Telecommunications Research Institute
**Dept. of Electronic Eng. Yonsei Univ.

ABSTRACT

In this paper, the performance simulator of the concurrent programmed internall call processing software of the TDX-10 electronic switching system is designed and implemented, while the performance is evaluated and analyzed by the simulator.

The standards of performnace evaluation are delay times of the grade of services recommanded in CCITT and the CPU busy rate.

As a result of this simulation, it is appreciated that the TDX-10 has the call processing capacity of 40,000 BHCA per ASS pairs. In addition, to improve the performance, new structures of the call processing software are presented and their perfomances are evaluated. The improved capacity of the internal call processing software becomes about 50,000 BHCA.

1. 서 론

교환기의 기본 기능은 역시 음성 교환 기능이며, 그 중에서도 호처리 소프트웨어의 구조는 전체적인 TDX-10 시스템의 성능에 매우 큰 영향을 주고 있으므로 호처리 소프트웨어의 성능에 큰 관심을 기울이지 않을 수 없다.

가입자에게 많은 종류의 양질의 서비스를 제공할 때에 분산 구조가 매우 유리하다. 따라서 분산 구조가 가지는 장점을 살리기 위하여 TDX-10 소프트웨어의 구조도 여러 개의 프로세서(processor)에 물리적으로 분산되어 있는 분산구조를 가지고 있다. 대용량의 분산 구조를 가지는 TDX-10의 호처리 소프트웨어의 성능평가는 매우 복잡하고 어렵다. 따라서 보다 종합적이고 실제 상황에 가까운 호처리 소프트웨어의 성능평가에 대한 연구의 필요성을 절실히 느낀다.

본 논문에서는 TDX-10 시스템의 현재 구현된 호처리 소프트웨어의 상세한 구조를 알아 본다. 그리고 호처리 소프트웨어의 성능평가를 위한 시뮬레이터(simulator)를 설계, 구현하여 발신음 지연과 호출음 지연을 기준으로 삼아 호처리 소프트웨어의 성능을 평가해 본다. 또한 이 시뮬레이터에서 생성되는 여러가지의 데이터를 토대로, 보다 더 나은 성능을 가진 호처리 소프트웨어를 설계하기 위한 기초자료를 산출하고자 한다.

2. 호처리 소프트웨어의 구조

호처리 소프트웨어는 4분야 8서브시스템에 걸쳐 여러 블럭으로 구성되며 각 블럭은 실제 실행 모듈로 구성되어 전문화된 기능을 담당한다. 하위 프로세서는 각 프로세서마다 하나의 블럭을 가지며 PPOS에의해 수행되며, 상위 프로세서는 여러개의 블럭을 가지며 각 블럭은 CROS의 도움으로 하나의 main process가 생성되어 블럭간 IPC(Inter Process Communication)에 의해 호를 제어한다.

사용자 기능 제어부의 주프로세스는 호가 시도되는 가입자 회선이나 트렁크마다 half call 프로세스를 생성하여 호를 제어하며, 서비스제어 계층과 signalling 정합 계층의 두 계층으로 이루어져 있어 새로운 기능의 추가를 용이하게 한다. 통화로계 정합부의 각 블럭은 하드에어를 제어하며 사용자 기능 제어부를 하드웨어의 구체적인 동작과 무관하도록 한다. 각 블럭은 하위계층의 프로세서에 실장되어 있으며 매우 실시간 처리를 요구하는 임무를 수행한다. 공통 통화로계 제어부의 블럭은 모두 주프로세스로 이루어져 있으며 통화로 접속과 신호의 송수신에 필요한 모든 경로의 채널을 배정하여 통화로계 정합부에 통화로계 하드웨어를 동작시키도록 함으로서 사용자 기능부가 통화로계에 무관하게 호를 제어하도록 한다. 번호번역부는 시스템의 번호 체계와

국간의 루팅, 망 구성 데이터를 구축하여 호의 진행에 필요한 착신 정보를 사용자 기능부에 제공하여 사용자 기능부에서 번호 체계나 망의 구성에 상관없이 호의 특성에 의해서만 호를 제어하도록 한다[12].

기본호에는 4가지형태의 호 즉 자국호, 출중계호, 입중계호, 중계호가 있다. 일반적으로 교환기의 호처리 성능의 평가는 4가지 혼합된 호를 처리하는 성능평가를 말한다. 그러나 여기서는 모든 호가 자국호라고 가정하고 이 자국호처리 성능을 측정하는 시뮬레이터를 설계하여 그 성능을 평가해 본다. 이 시뮬레이터는 나아가 4가지 형태의 호를 처리할 때의 성능을 측정하는 궁극적인 시뮬레이터 설계의 토대가 될 것이다.

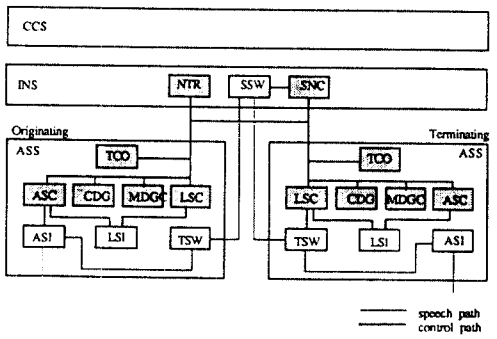

그림 2-1. 자국호 관련 블럭의 역할

3. 자국호 성능평가 시뮬레이터의 설계

순차적 프로그래밍에의해 구현된 교환기의 호처리 소프트웨어의 성능을 평가한 연구는 문헌에서 볼 수 있다[13]. 그러나 동시적 프로그래밍에의한 호처리 소프트웨어의 성능평가는 이와는 다르게 이루어져야 한다. 동시적 프로그래밍 기법은 응용 프래그래머의 소프트웨어 생산성을 상당히 높여주나 그만큼 OS에 부하를 주게 되어 OS가 매우 복잡하다[14]. TDX-10의 호처리 소프트웨어는 소프트웨어 모듈성(modularity)를 위하여 각각 자신의 독립적인 목적을 가지는 여러개의 소프트웨어 블럭들로 구성되어 있다. 이들 블럭은 프로세스라는 단위로 이루어져 있다. 호처리 소프트웨어는 호의 흐름을 제어 하기위하여 프로세스간에 메시지 교환을 한다. OS는 프로세스를 수행의 기초단위로 관리하므로 IPC에 의한 메시지 교환을 빈번히 수행하고 이에따라 콘텍스트스위칭이 자주 일어나게 된다. 그러므로 OS가 콘텍스트스위칭, 외부 IPC 처리를 위한 인터럽트 그리고 RTC 인터럽트를 처리하는 알고리즘을 반영해야 한다. 그 밖의 OS 메니저(manager) 특히 메모리 메니저에의해 소비되는 시간은 호처리 프로그램 수행 시간에 포함시켜 성능 측정에 사용한다.

한편 통계적인 가입자의 습성을 나타내는 호의 형태를 보면 다음과 같다.

- 다이얼을 돌리지 않고 끊내는 호가	5 %
- 다이얼링 도중에 끊내는 호가	5 %
- Network Busy 호가	5 %
- 착신 통화중인 호가	15 %
- 착신 무응답인 호가	5 %
- 통화가 이루어지는 호가	60 %

그리고 완전히 착신 번호를 누르는데 걸리는 시간, 즉 DTMFR 평균 보류시간이 7.2 초이고 자국 통화보류 시간이 63 초이며 평균 링이 울리는 시간은 15 초이다. 이러한 통계값을 시뮬레이터에 그대로 반영하였다. 단 network busy는 DTMFR과 switch busy에 의해 일어나는 것으로 간주하여 이들이 busy가 되지 않는한 network busy는 일어나지 않는 것으로 간주하였다.

기본적인 생각은 호를 일정시간 간격으로 계속적으로 발생시키고 발생된 호는 호의 시작 단계에서 부여한 호의 호름에 따라 처리된다. 이 때 응용 프로그램의 수행 시간을 계산하여 호의 처리를 지연시켜 1 시간 동안 몇 개의 호를 처리하는지를 알아보고 발신음과 호출음의 지연이 CCITT에서 권고한 기준에 알맞는 지를 조사한다. 또한 호처리 소프트웨어의 구조를 변경시켰을 때 이에따른 시스템의 성능변화가 어떻게 변하는지를 측정하고자 한다.

시뮬레이터의 전체적인 구조는 부록의 그림A.1의 SDL 다이어그램과 같다. 이 시뮬레이터는 ASP를 중심으로 여기에 관련된 각 PP와 NTP 그리고 INP의 메시지 처리및 IPC를 고려하여 설계하였다. TDX-10에는 ASS가 60개까지 존재할 수 있으나 1개의 ASP에 8,000 가입자가 수용될 때 처리해야 하는 호처리 용량인 40,000 BHCA의 half call을 처리할 수 있는 지를 측정하기 위하여 100,000 가입자가 30개의 쌍을 이루어 호의 발착신이 이루어지는 것으로 가정하였다. 그러므로 1개개의 ASP에는 발신호와 그 발신호에의해 착신되는 만큼의 착신호가 동시에 존재하는 것으로 생각한다. 또한 1개의 ASP에서 1호를 처리하는 것은 NTP나 INP에서는 60호를 처리하는 부하가 걸리는 것으로 간주하였다. 발생된 호는 각기 다른 ASP에 있는 가입자간에 이루어지는 호로 간주하였다. 즉 모든 호가 공간스위치를 통하여 이루어지고 intra-junctor를 통한 호는 없다고 가정하였다.

먼저 simulator의 구조를 보면 TU_dec 프로시쥬어에서는 단위시간 10 μsec로 감소시키면서 NTP, INP, ASIP, LSP, ASP에서 들어오는 IPC가 있는지 조사하고 RTC 인터럽트와 IPC 인터럽트의 발생과 호의 발생을 시간에 따라 생성시켜 주는 부분이다. RTC_dec 프로시쥬어는 RTC 인터럽트 처리시에 걸리는 시간을 지연시키고, XIPCI_dec 프로시쥬어는 외부 IPC가 들어와 6 msec 마다 발생한 인터럽트나 외부 메시지가 6 msec 이내에 4개가 도착하면 발생하는 인터럽트의 처리를 하는데 걸리는 시간을 지연시킨다. PROCESS_dec 프로시쥬어는 OS의 프로세스 메니저(process manager)의 프로세스 스케쥴링 정책에 따라 프로세스를 스케쥴링하는 부분이다. 그리하여 콘텍스트스위칭이 일어나면 이 때 소비되는 시간만큼 호처리를 지연시킨다. 사용자의 요구에 의하여 생성된 프로세스는 dormant 상태에 있다가 수행대기(ready) 상태가 되고, 디스팻쳐(dispatcher)에 의해 수행중(running) 상태로 천이되어 동작한다. 이후에 원하는 목적 및 기능을 수행하기 위하여 각종 blocked 상태로 천이되기도 하는데, 후에 다른 프로세스들에 의해 재활성(reactivate)되어 다시 수행대기 상태로 천이된다. 이러한 과정들을 거쳐 수행이 완료되면 종료(terminate) 상태로 천이된다. 대부분의 호처리 프로세스는 수행중 상태에서 그 동안 수신된 메시지를 처리하고 더이상

처리할 메시지가 없거나 RECEIVE CASE 문을 만나서 이미 들어온 메시지는 존재하나 그 상태에서 처리하는 메시지가 없으면 receive_block 상태로 천이된다. 수행대기 상태의 프로세스는 수행대기 큐에 들어온 순서에 따라 FIFO(First In First Out)로 CPU 제어를 받는다. TDX-10의 프로세서는 MC68020이나 MC 68030이며 이들은 두 가지 모드로 동작된다[8][9][10]. Supervisor mode에서 동작중인 프로세스는 RTC(Real Time Clock) 인터럽트가 걸려도 preemption 당하지 않으나 user mode에서 수행중에 RTC 인터럽트를 만나면 preemption 당하여 CPU 제어를 우선순위가 가장 높은 프로세스에 넘겨준다. 따라서 CROS의 프로세스 스케쥴링 정책은 우선순위가 있는 FIFO 구조를 가진다. APPLICATION_dec 프로시쥬어는 수행중 상태에 있는 프로세스의 응용 프로그램이 수행하는 각 호처리 절차에 따라 호처리 흉내를 내는 부분으로 호처리 소프트웨어와 같은 구조를 가진다. 여기서는 호처리 응용 프로그램이 메시지를 처리하는데 걸리는 시간만큼 호처리를 지연시켜준다. 응용 프로그램에서 걸리는 수행 시간의 처리는 실제의 호처리 프로그램과 같게 만들어진 여러 프로시쥬어 즉 ASC_main, ASC_orig, ASC_term, TCO_main, TCO_subs, LSC_main, MDGC_call, CDG_init, CDG_fin에서 수행된다. 시뮬레이터를 구현하는데 사용한 언어는 TDX-10 호처리 소프트웨어의 구현 언어인 CHILL과 정합을 위하여 CHIIL 언어를 사용하였다.

4. 자국호의 성능 평가 및 분석

4.1 자국호의 성능평가

이제 구현된 자국호 성능평가 시뮬레이터를 이용하여 시스템의 호처리 용량을 측정해 본다. 먼저 각 호처리 블럭에서 처리되는 메시지의 처리시간을 산출해야 한다. 메시지 처리시간의 산출은 한 호당 걸리는 처리시간을 TDX-10 시험모델 교환기에서 추출한 다음, 호처리 프로그램의 OS 프리미티브 수행시간, 프로세스의 생성 및 소멸 시간, DB 접근 시간 그리고 순수 호처리 프로그램 수행시간을 고려하여 계산된 값이다. 각 메시지 처리시간은 다소 오차를 가지나 전체적인 수행시간은 실제로 측정된 값이다[15].

교환기의 성능평가는 일반적으로 발신음 지연시간과 호출음 지연시간으로 평가되며 프로세서의 CPU 점유율이 90%에 이르면 과부하 제어를 하게 됨으로 CPU 점유율은 매우 중요한 성능평가 자료가 된다.

- 서비스 기준 지연시간

CCITT Q543에서 권고된 서비스등급 지연시간의 허용치는 아래의 표4-1과 같다.

표 4-1. CCITT 권고 서비스기준 지연시간

(msec)

서비스기준 항목	기준부하 A		기준부하 B	
	평균	95%	평균	95%
발신음 지연	400	600	800	1000
호출음 지연	650	900	1000	1600

- 발신음 지연시간

발신음 지연시간은 교환기의 가입자 회선 정합장치에서 hook_off를 인지한 순간부터 교환기가 발신음을 송출할 때까지의 지연시간이다. 그림4-1에는 자국호 처리과정중에서 발신음 지연시간에 관련된 부분을 처리하는 호처리 소프트웨어의 호처리 순서도이다.

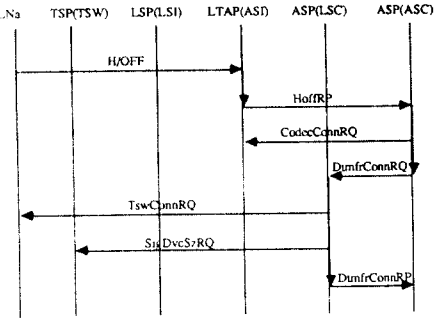

그림 4-1. 발신음 지연 관련 호처리 순서도

- 호출음 지연시간

호출음은 교환기가 가입자로부터 착신번호의 마지막 디지트를 인지한 후 발신 가입자에게 호출음을 송출 할 때까지 걸리는 시간을 말한다. 그림4-2에 호출음 송출 지연시간에 관련된 부분을 처리하는데 소요되는 호처리 순서도를 나타내고 있다.

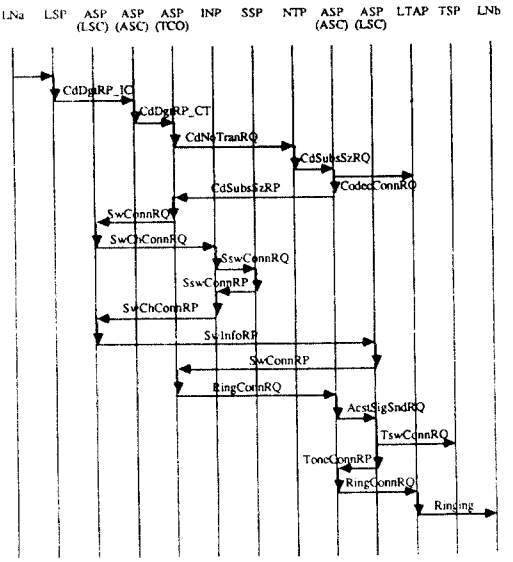

그림 4-2. 호출음 지연 관련 호처리 순서도

4.2 결과 분석

얻어진 결과를 그림4-5의 (1)에 표시하였다. 이 결과를 분석해 보면 발신음 지연은 시스템에 가해지는 부하에 상관없이 비교적 거의 일정함을 볼 수 있다. 이것은 시스템이 분산 구조를 가지는 결과로 해석되며 전 시스템에 가해지는 부하에 상관

없이 자신의 ASP 부하에 따라 결정된다. 호출음은 이와달리 집중화된 INS에 따라 시스템의 전체적인 부하에 민감한 영향을 받는다. 따라서 전 시스템의 부하가 증가함에 따라 NTP와 INP부하가 증가하여 호출음 지연시간에 큰 영향을 미치므로 호출음 지연시간도 급격히 증가됨을 볼 수 있다.

ASP의 CPU 점유율이 90%에 이를 때 그 ASS는 과부하 제어를 하게되며, CPU 점유율이 90%에 이를 때의 호처리 용량을 각 ASS의 호처리 능력으로 삼는다. 그러므로 그림6-3에 나타난 것처럼 1 ASS는 4만 BHCA 정도의 호를 처리하는 것으로 판정된다. 1 ASS당 4만 BHCA의 호처리 용량은 각 ASS가 처리해야하는 호처리 용량에 겨우 미치는 정도로 호의 발생이 상당히 폭주하게 되면 더 이상의 호처리 성능을 내지 못하므로 호가 poisson 분포로 발생한다고 가정할 때 서비스받지 못하는 호가 상당히 증가되어 전체의 호성공율이 60%에 이르지 못할 것으로 생각된다. 따라서 더 높은 호처리 능력을 가져야만 양질의 서비스를 할 수 있게 된다. 교환기의 호처리 성능을 올리려면 호당 수행시간을 줄여야 한다. 한 호당 수행시간을 줄이려면 호처리 소프트웨어가 새로운 구조를 가져야함은 당연하다.

4.3 새로운 호처리 소프트웨어의 구조의 제안 및 성능평가

TDX-10의 호처리 소프트웨어는 여러개의 독립적인 블럭으로 구성되고 동시적 프로그래밍 기법을 사용하고, 블럭간의 메시지 교환은 IPC를 사용하며, 공통 데이터는 DB를 구축하여 DBMS를 통하여 접근한다. 그러므로 수행시간의 대부분을 블럭간 메시지전달, OS 프리미티브 사용, DB 접근 그리고 프로세스의 생성 및 소멸이 차지하므로, 한 호당 수행시간을 줄이는 방법은 블럭간 내부 메시지를 줄이는 방법, OS 프리미티브의 사용을 줄이는 방법, DB 접근 횟수를 줄이는 방법, child process의 생성 및 소멸을 줄이는 방법등이 있다. 이 중에서 가장 실현가능한 방법은 내부 IPC를 줄이는 방법이다. 그림4-1의 구조에서 IPC를 줄이는 방법은 가입자 프로세스와 트렁크 프로세스로 구성되어 있는 블럭 TCO를 분해하여 각각의 프로세스를 ASC와 TKC로 통합시켜 ASC와 TCO간, TKC와 TCO간의 메시지를 줄이는 것이다. 이와 관련된 블럭의 구조를 그림4-3에 도시하였다. 이 경우에 줄어드는 메시지는 호당 13개가 된다. 편의상 기존의 호처리구조를 호처리구조1이라 하고 새로운 구조를 호처리구조2라고 부르기로 한다. 한편 MDGC는 호의 통계를 수집하는데 호의 여러 국면마다 내부 IPC를 통하여 데이터를 수집하고 있다. 각종 통계를 해당 블럭에서 수집하여 처리하도록 하면 호당 17개의 메시지가 줄어든다. 이 때의 호처리구조를 호처리구조3이라 하고 그 구조가 그림4-4에 있다. 호처리구조2는 호처리구조1보다 13개의 메시지가 줄어 들며 따라서 호당 수행시간도 줄어든다. 또한 호처리구조3에서는 전체적으로 줄어드는 메시지의 갯수가 30개이며 호처리구조1이 호당 97개의 메시지를 처리하는 것을 고려하면 메시지가 상당히 줄어든 구조이다.

그림 4-4. 호처리구조3의 자국호 관련 블럭의 역할 새로운 구조의 메시지 처리시간을 시뮬레이터에에 입력시켜서 시뮬레이션을 수행한 결과가 그림4-5에 나타나 있다. 발신음 지연시간이 급격히 상승하는 시점이 이전의 구조보다 더 큰 값의 BHCA에서 나타났으며, 호출음 지연 시간도 비슷한 모습을 보여 주었다. 한편 CPU 점유율을 살펴보면 90%의 CPU 점유율이 48,000 BHCA와 52,000 BHCA에서 나타나 이전의 구조보다 약 25%의 성능개선 효과를 보였다. 그러나 새로운 구조는 소프트웨어 modularity가 앞의 구조보다 떨어져서 디지틀 가입자나 ISDN 가입자의 호를 처리하기 위해 필요한 디지틀 가입자 블럭 및 No.7 signalling을 하는 국간의 호를 처리하기 위한 블럭의 추가를 고려할 때, 프로그램의 중복을 가져올 것으로 생각된다.

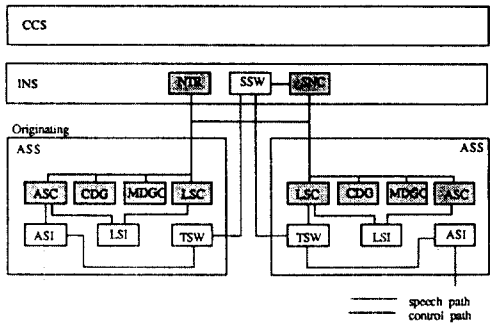

그림 4-3. 호처리구조2의 자국호 관련 블럭의 역할

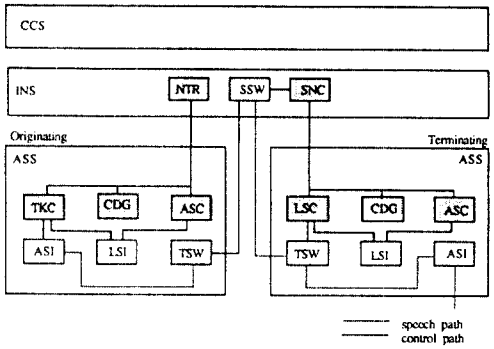

그림 4-4. 호처리구조3의 자국호 관련 블럭의 역할

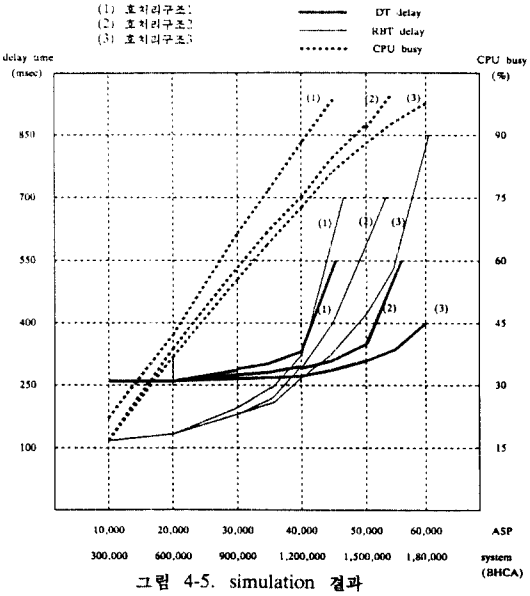

그림 4-5. simulation 결과

5. 결 론

본 논문에서 구현된 시뮬레이터는 TDX-10의 소프트웨어 버전4(version 4)의 호처리 소프트웨어 자국호 성능을 평가하기 위한 것이다. 시뮬레이터는 호처리 응용 프로그램의 메시지의 처리시간과 메시지가 IPC 입력 큐에서 대기하는 대기시간 그리고 OS의 기본적인 수행시간을 반영하도록 설계되었으며 실험적으로 측정하기 어려운 시간요소는 호처리 프로그램 수행시간에 포함시켰다.

시뮬레이션 수행결과 하나의 ASS는 약 40,000 BHCA의 호를 처리할 수 있는 성능을 가진 것으로 판정되었다. 더욱 큰 성능을 가지도록 하기 위해 새로운 호처리 소프트웨어의 구조를 제안하였다. 또한 제안된 호처리구조의 성능을 평가해 보았다. 호당 생성되는 process의 갯수를 3개에서 2개로 줄이는 구조를 가지도록 하고 통계 데이터의 수집을 호처리 각 블럭에서 하도록 호처리 소프트웨어의 구조를 바꾸면 약 50,000 BHCA 이상의 호처리 성능을 내는 것으로 판정되었다. 이것은 수치적으로 25%의 성능 향상을 말한다. 그대신 소프트웨어 모듈성(modularity)은 많이 떨어질 것으로 예상되며 이로 인하여 블럭들간의 프로그램이 중복되는 현상을 가져오게 될 것이다. 그러나 교환기의 성능은 매우 중요한 요소이며 교환기 개발의 성공 여부를 판정해주는 요소이므로 더 높은 성능의 교환기가 요구된다.

본 논문은 이와같은 관점에서 교환기의 성능평가를 시도하였고 더 높은 성능을 가지는 호처리 소프트웨어를 제안하고 그것의 성능을 평가함으로서 개발중에 있는 TDX-10 전자교환기의 성공적인 개발에 이바지 할 수 있으리라 생각한다.

4가지 형태의 호처리를 동시에 수행하는 호처리 성능의 평가를 위한 시뮬레이터의 설계가 궁극적인 목표인데 본 논문에서 구현된 시뮬레이터를 출중계호, 입중계호 그리고 중계호의 호처리 과정을 처리할 수 있도록 변경하면 목표하는 시뮬레이터를 구현할 수 있을 것이다.

호처리 프로그램의 수행시간값의 변동에 따라 측정되는 호처리 용량이 변하기 때문에 보다 정확한 산출이 요구된다. 그러므로 호처리 프로그램이 메시지를 받아서 처리하는 수행시간을 정확히 산출하는 연구가 병행되어야 할 것이다.

참고 문헌

[1] 한국전자통신연구소, "전전자 교환기 연구개발 총괄 보고서", 1989.12.

[2] 한국전자통신연구소, "TDX-10 시스템구조도", 1990.9.

[3] 한국전자통신연구소, *"Cocurrent Realtime Operating Systems"*, 1988.9.

[4] 한국전자통신연구소, *"Concurrent Realtime Operating System User Manual"*, 1988.4.

[5] 한국전자통신연구소, *"Peripheral Processor Operating System"*, 1988.9.

[6] 한국전자통신연구소, *"CHILL Reference Manual"*, 1988.8.

[7] 한국전자통신연구소, *"CHILL Programming Guide"*, 1990.7.

[8] MOTOROLA, *"MC68020 User Manual"*, Prentice-Hall, Inc., 1985.

[9] MOTOROLA, *"MC68030 User Manual"*, Prentice-Hall, Inc., 1990.

[10] Thomas L. Harman, *"The Motorola MC68020 and MC68030 Microprocessor"*, Prentice-Hall International, Inc., 1989.

[11] 한국전자통신연구소, *"TDX-10 Data Base Management System"*,1989.4.

[12] 안시환외 4인, "TDX-10 호처리 소프트웨어 구조", 한국통신학회 1990년도 하계종합학술발표회 논문집, 1990.8, pp. 875 ~ 879.

[13] Mischa Schwartz, *"Telecommunication Networks"* : Protocols, Modeling And Analysis", Addison-Wesley Publishing Company, 1987, pp. 567 ~ 594.

[14] Gregory R. Andrews and Fred B. Schneider, "Concepts and Notations for Concureent Programming", *Computime Surveys*, Vol. 15, No.1, March 1983, pp. 1 ~ 43.

[15] 류병한, "시뮬레이션에 의한 TDX-10 제어계 성능분석", 한국전자통신연구소 문서, TT/R-901005, 1990.10.

[16] Grinsec, *"Electronic Switching"*, Elservier Science Publishers B.V., 1983 pp. 313 ~ 453.

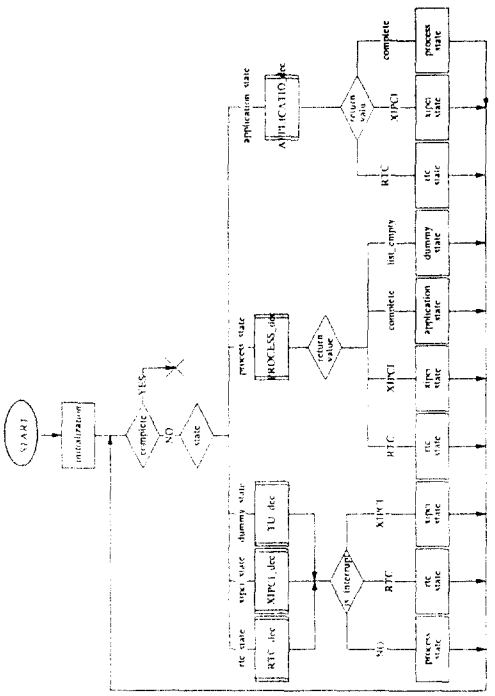

그림 A-1. simulator의 SDL diagram