이동통신시스템의 제어국 프로세서를 위한 실시간 사용자 지원 시스템의 설계 및 구현

(Design and Implementation of Real-time User Support System for Base Station Control Processor in Mobile Communication Systems)

박 우 구 † 이 제 헌 ††
(Woogoo Park) (Jehun Rhee)

요 약 호제어 프로세서와 같은 호제어 프로세서는 사용자의 다양한 요구 사항을 제공해야 한다. 그런데 몇몇의 프로세서들은 아직도 embedded 보드 형태로 호처리나 운용 및 유지보수 기능을 수행하는 실정이다. 이와 같은 형태로는 사용자에 의해 행해지는 시험이나 디버깅과 같은 다양한 환경을 제공할 수 없다. 대부분의 시스템에서 시스템에 대한 제일 중요한 고려 사항으로서 사용자 환경이 중요하게 여겨지고 있으며, 특히 이동통신 시스템에서는 호제어 프로세서의 설계 및 구현시에 더욱 그러하다. 본 논문에서는 호처리 및 유지보수 기능을 향상시키는 QUEST (QNX-based User Environment Support Tool) 시스템을 설계 및 구현하였다. 시뮬레이션을 통하여 제안한 시스템의 성능을 분석한 결과 기지국 제어기의 성능에 큰 영향을 미치지 않고 동작하는 것을 확인하였다.

Abstract Processors for a base station controller such as call control processor have to serve users demand. Some of processors have been still used for the processing of call and O&M (operation and maintenance) management by adopting embedded board type. This type can not provide users with a wide variety of environments for testing and debugging. User environments are regarded as a primary system factor in most systems and this is particularly so in designing and implementing the processor for base station controller in mobile communication systems. We describe the design and implementation of new user environment tool named QUEST for the next generation mobile communication system, IMT-2000. The simulation results of performance evaluations demonstrate that our system performs well for a base station controller without performance degradation.

1. 서 론

이동통신 가입자의 급속한 증가에 따라 많은 기지국이 설치되고 있으며, 기지국의 증가는 전력 제어 및 발신호, 착신호, 핸드오프 호의 처리 및 각종 운용 및 유지보수를 위하여 기지국 제어기가 증가된다. 기지국 제어기는 호 제어 프로세서에 의해 위의 여러 가지 기능들을 수행한다. 호처리와 운용 및 유지보수 기능의 효율적인 처리를 위하여 운용자는 기지국 제어기에서 많은 디버깅 시간을 필요로 한다. 이와 같이 디버깅을 위한 쉘 도구가 CROS (Concurrent Realtime OS)라는 실시간 운영체제에 선보인 이래 많은 운용자가 이를 이용하였다[1] [2]. 그러나 이동통신의 환경 및 특성, 사용자의 다양한 요구, 사용하는 정보의 다양성 및 ATM과 같은 전송 매체의 추가에 따라 기존의 시스템이 갖는 제한적인 기능은 여러 가지 측면에서 한계를 드러내고 있다. 따라서 시그널 모니터링과 사용자 접근 권한 제어 그리고 다양한 전송 매체에 대한 지원 등의 요구에 따라 IMT-2000에서는 새로운 기능을 갖는 실시간 사용자 지원 시스템을 필요로 하게 되었다[3] [4]. 이와 같은 사용자의 요구에 맞도록 QUEST라는 실시간 사용자 환

† 정 회 원 : 한국전자통신연구원 무선방송기술연구소 연구원
wgpark@etri.re.kr
†† 비 회 원 : 한국전자통신연구원 무선방송기술연구소 연구원
jhrhee@etri.re.kr
논문접수 : 1999년 3월 12일
심사완료 : 1999년 10월 15일

경 지원 시스템이 QNX라는 실시간 운영체제하에서 설계 및 구현되었다[5] [8]. QUEST는 크게 2가지의 기능을 갖도록 설계되었다. 하나는 사용자와의 내부 인터페이스로서 사용자에게 다른 시스템과의 통신을 통합된 방법에 의해 송수신이 가능하도록 사용자 인터페이스를 제공하는 기능이고, 또 다른 하나는 쉘(Shell) 환경을 제공하여 시그널(혹은 메시지)의 인위적인 송수신, 시그널 추적, 자동 전송, 시나리오에 의한 시그널 모니터링, 터미널 상에 송수신되는 시그널의 출력을 가능/억제하는 toggle 기능, 사용자가 제공하는 특정 파일로 송수신되는 시그널의 정보를 저장하여 제공하는 로깅(logging) 기능과 각종 toggle 및 로깅 기능의 정보를 검색하는 view 기능, 프로세스 관리 기능, 파일 관리 기능, 통계 및 측정 기능, 사용자 접근 권한 제어 기능, 명령어 제어, 그리고 이력, 도움말, 부팅 및 로딩, 버전 관리, 시스템 정보 변경 기능 등으로 구성되어 있으며, 위의 각종 쉘 환경 명령어는 명령어 풀(pool)을 이용하여 쉽게 등록 및 삭제가 가능하게 하여 시스템의 사용의 편이성을 도모하였으며, 이러한 모든 명령어들은 명령어 풀을 통하여 임의의 사용자에 의해 쉽게 접근할 수 있도록 설

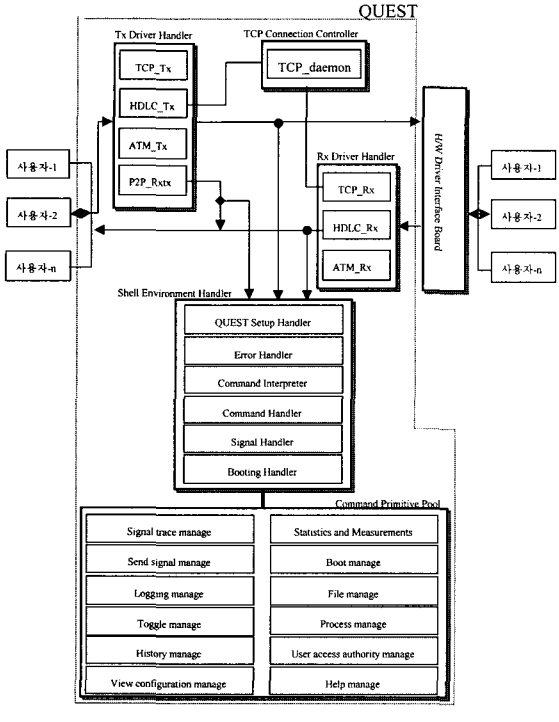

그림 1 QUEST 구조

계되어 있다. QUEST의 장점은 다음과 같다.

- 통합된 사용자 인터페이스 제공
- 지원 환경의 개방성 및 투명성 제공
- 드라이버 변경, 추가 및 제거의 용이성
- 다양한 쉘 명령어 프리미티브 제공
- 쉘 명령어 프리미티브의 독립성 보장
- 손쉬운 쉘 명령어 프리미티브의 추가 및 삭제
- 사용자 접근 권한 기능 부여에 따른 신뢰성 향상

본 논문의 구성은 다음과 같다. 제 2 장에서는 설계 측면에서 본 QUEST의 전체적인 구조에 대하여 살펴보고, 제 3 장에서는 QUEST의 구현상의 기능적 특징에 대해서 살펴보고 제 4 장에서는 본 실시간 사용자 환경 지원 시스템을 이용하여 시그널을 전송했을 때의 전송 속도에 따른 성능 분석을 실시하였고, 마지막 장에서 결론을 맺었다.

2. QUEST의 설계

2.1 QUEST의 구조

QUEST는 모듈화 및 개방성에 근거하여 다음과 같은 4부분으로 구성한다. 전체 구조는 그림 1과 같으며 제 3 장에서 자세한 기능을 설명한다.

- Tx 드라이버 처리기 (Tx Driver Handler)
- Rx 드라이버 처리기 (Rx Driver Handler)
- 쉘 환경 처리기 (Shell Environment Handler)
- 명령어 프리미티브 풀 (Command Primitive Pool)

2.2 디렉토리 구성

QUEST 시스템이 운용 중에 접근하는 디렉토리는 크게 4가지로 구성된다. 먼저, QUEST 시스템이 동작하는 디렉토리로 QUEST와 함께 실행되는 모든 사용자 프로그램이 같이 상주하고 있다. 이 밖에도 입출력 시그널에 대한 logging시 시그널 정보를 저장하는 logging 파일(**파일.log**)이 이 디렉토리에 저장된다. 그리고 **LOG** 디렉토리는 사용자 로그인 생성시 (명령어 **qsu**) 본 디렉토리에 생성시각.sys 형태로 저장한다. 또한, 메뉴 정보를 저장하고 있는 **MENU** 디렉토리와 명령어를 저장하고 있는 명령어 풀인 **POOL** 디렉토리 등으로 구성되어 있다.

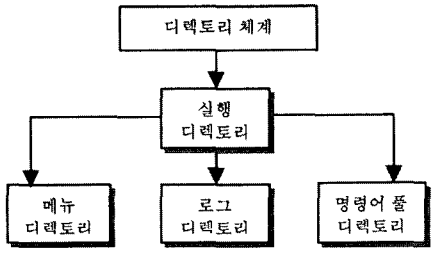

그림 2 디렉토리 구성

2.3 데이타 구조

QUEST는 다수의 프로세스가 상호 작용을 통하여 매핑 되어 있어 서로 동일하게 사용하는 데이타들에 대한 구성을 공유 메모리(shared memory)를 이용해서 구현한다. 데이타는 크게 일반 제어 데이타, 시그널 추적용 데이타, Toggle 및 Logging 데이타, 프로세스 관리 데이타, 파일 관리 데이타, 그리고 사용자 접근 권한 제어 데이타 등으로 구성되어 있다. 각 데이타에는 통계 수집을 위하여 시도 수 오류 수 그리고 성공 수와 같은 정보가 저장되어 있다. 각 데이타의 종류 및 기능은 다음과 같다.

2.3.1 일반 제어 기능 관련 공유 데이타 구조

일반 제어 기능을 위한 데이타에는 이력(history) 제어, 도움말 제어, 부팅 제어, Pool 검색 제어, 탈출, 버전 검색, 시스템 정보 갱신 등이 있다.

표 1 일반 제어용 데이타 구조

데이타 분류	구조		형식
이력 제어	최대 제어 가능 이력 수		Short
	등록된 이력 수		Short
	이력 별	이력 명	Char string
		이력 입력 시각	Char string
		사용자 로그인 명	Char string
도움말 제어	최대 제어 가능 도움말 수		Short integer
	등록된 도움말 수		Short integer
	도움말 별	명령어 명	Char string
		명령어 사용법	Char string
		명령어 예	Char string
		버전 번호	Char string
		시도 수 (통계)	Unsigned int
		성공 수 (통계)	Unsigned int

2.3.2 시그널 제어 기능 관련 공유 데이타 구조

시그널 제어 기능을 위한 데이타에는 시그널 추적을 위한 기능 등록, 시그널 추적 활성화, 시그널 추적 비활성화, 시그널 축적 기능 해지, 자동 시그널 시험, 시그널 전송 정보 출력, 시그널 추적 정보 검색, 시나리오에 의한 시그널 모니터링 기능 등록, 해지 및 검색 등이 있다. 다음의 데이타는 시그널 추적 제어를 위한 공유 데이타 구조를 나타낸 것이다.

표 2 시그널 추적용 데이타 구조

구조		형식
최대 시그널 제어 가능 수		Short
시그널 감시 주기		Short
최대 시그널 활성화 수		Short
입력 시그널 등록 수		Short
출력 시그널 등록 수		Short
입출력 시그널 별	프로세스명	Char string
	프로세스 ID	Process Instance ID
	시그널 ID	Integer
	출력 모드	Head|body
	추적 모드	PID-level|SID-level
	트리거 시간	Short
	저장용 트리거 시간	Short
	활성화 시간	Short
	저장용 활성화 시간	Short
	추적 상태	Act|deact|idle

2.3.3 Toggle 및 Logging 제어 기능 관련 공유 데이타 구조

Toggle 및 Logging 제어 기능에는 Toggle 기능의 활성화 및 비활성화, Logging 기능의 활성화 및 비활성화, 그리고 Toggle 및 Logging 상태 출력 등이 있다.

2.3.4 사용자 접근 권한 제어 기능 관련 공유 데이타 구조

사용자 접근 권한 제어 기능에는 사용자 접근 권한 제어, 시스템 제어, 사용자 로그인 정보, 사용자 로그 관리 등이 있다.

2.3.5 명령어 제어 기능 관련 공유 데이타 구조

명령어 제어 기능에는 명령어 허가 및 금지 기능과 명령어 예약 기능이 있다. 다음의 데이타는 명령어 제어를 위한 공유 데이타 구조를 나타낸 것이다.

표 3 Toggle 및 Logging용 데이타 구조

구조		형식
Logging 파일 Pointer		FILE *fp
Logging 파일 명		Char string
Logging 상태		Short
Toggle 상태		Short
최대 logging 파일 부가 수		Short
등록된 logging 파일 부가 수		Short
최대 측정 시그널 수		Integer
등록된 측정 시그널 수		Integer
시그널 별	시그널 ID	Integer
	수신 시그널 수	Unsigned int
	수신 시그널 시각	Clock t
	수신 시그널간 도착 시간	Double
	송신 시그널 수	Unsigned int
	송신 시그널 시각	Clock t
	송신 시그널간 도착 시간	Double
	측정 모드	ON|OFF
	시각 모드	MEAS_SEC| MEAS_MIN| MEAS_HOUR
	분포 형태	MEAS_NORMAL| MEAS_POISSON| MEAS_EXP
	바로 전 측정 시 수신 시그널 수	Unsigned int
	저장된 수신 시그널간 도착 수	Double
	저장된 측정 시간	Unsigned short
	시 표식	Unsigned int
	분 표식	Unsigned int
	초 표식	Unsigned int
	지속 시간	Unsigned short
	최대 감시 주기	Short

표 4 명령어 예약 제어용 데이타 구조

데이타 분류	구조		형식
명령어 허가 및 금지	최대 금지 명령어 등록 가능 수		Short integer
	등록된 금지 명령어 수		Short integer
	명령어 별	명령어 명	Char string
		동작 형태	CMD_PLUS_ADD| CMD_MINUS_REM
명령어 예약	최대 명령어 예약 가능 수		Short
	예약된 명령어 수		Short
	명령어 별	활성화 시간 (시)	Unsigned int
		활성화 시간 (분)	Unsigned int
		활성화 시간 (초)	Unsigned int
		사용자 로그인 명	Char string
		명령어 명	Char string
		예약 상태	REG_WAIT| REG_ACTIVE| REG_IDLE| REG_CANCEL| REG_SUSPEND

이 외에도 프로세스 및 파일 제어용 공유 데이타가 존재한다.

3. QUEST 구현상의 기능적 특징

본 장에서는 제 2 장의 QUEST 설계 측면의 구조에 따라 다음과 같은 Tx 및 Rx 드라이버 처리기와 쉘 환경 처리기를 구현하고 구현측면에서 각 처리기의 기능적 특징을 살펴보았다.

3.1 Tx 드라이버 처리기 (Tx Driver Handler)

QUEST에서는 사용자의 편리성과 친숙성 및 클라이언트로 송신하는 시그널 또는 메시지의 내부 정보를 용이하게 파악하기 위하여 프로토콜별로 드라이버 처리기를 드라이버로부터 분리하여 독립화 시킨다. 따라서 드라이버별로 특성을 갖는 정보나 조정 과정은 해당 드라이버 처리기에서 수행하고 실질적인 데이타나 메시지의 송신에 관한 사항은 상호 연결된 드라이버 제어기(driver controller)에서 분담하여 처리한다. 드라이버 처리기는 3.3의 쉘 환경 처리 부분과 연동하여 시그널 및 메시지의 내부 정보를 검색하여 사용자에게 제공한다.

3.2 Rx 드라이버 처리기 (Rx Driver Handler)

QUEST에서는 사용자의 편리성과 친숙성 및 클라이언트로부터 수신되는 시그널 또는 메시지의 내부 정보를 용이하게 파악하기 위하여 프로토콜별로 드라이버 처리기를 드라이버로부터 분리하여 독립화 시킨다. 따라서 드라이버별로 특성을 갖는 정보나 조정 과정은 해당 드라이버 처리기에서 수행하고 실질적인 데이타나 메시지의 수신에 관한 사항은 상호 연결된 드라이버 제어기(driver controller)에서 분담하여 처리한다. 드라이버 처리기는 3.3의 쉘 환경 처리 부분과 연동하여 시그널 및 메시지의 내부 정보를 검색하여 사용자에게 제공한다.

3.3 쉘 환경 처리기 (Shell Environment Handler)

사용자는 QNX상에서 다른 시스템과 통신하고자 할 때 송수신되는 시그널 또는 메시지 정보를 편리하게 획득 가능해야 하며, 다른 프로세스(peer-to-peer)에게 시그널 및 메시지를 전송하여 보다 융통성 있는 사용자 환경을 제공해야 한다. 이 밖에도 on-line 상태에서 시그널 및 메시지를 억제/가능 하게 하거나 또는 로그 파일에 저장하는 등의 다양한 지원 환경을 제공해야 하는 것은 필수적이다.

QUEST는 이와 같은 다양한 사용자의 욕구를 실시간 적으로 만족시키기 위해서 쉘 환경을 구축하는 것이 바람직하다. QUEST에서 지원하는 쉘 환경은 on-line 상에서 다음과 같은 기능을 제공하도록 설계된다.

- QUEST 초기화 처리기 (QUEST Setup Handler)
- 오류 처리기 (Error Handler)
- 시그널 처리기 (Signal Handler)
- 명령어 해석기 (Command Interpreter)
- 명령어 처리기 (Command Handler)
- 부팅 처리기 (Booting Handler)

3.3.1 QUEST 초기화 처리기 (QUEST Setup Handler)

본 기능은 프로세서 시동 및 재시동 기능에 의해 동작한다. 여기서는 시스템 인터럽트 처리를 위한 시그널 등록 및 각종 데이타의 초기화, 사용중인 공유 메모리(shared memory)를 해제하여 초기화시킨다. 또한, HDLC pci334 보드로 초기 데이타를 다운 로드시키는 등의 작업을 수행한다. 마지막으로 HDLC Tx/Rx 드라이버 처리기 프로세스를 생성시키고 사용자로부터의 쉘 명령어를 받아들이기 위하여 쉘 환경 처리기 중 명령어 해석기(command interpreter)를 호출한다. 세부 기능은 다음과 같다.

- 시스템 인터럽트 처리를 위한 시그널 등록
- Tx/Rx 드라이버 처리기의 소멸 인식을 위한 시그널 등록
- 공유 메모리의 해제
- 시그널 정보 오류 카운터의 초기화
- HDLC 데이타 초기화
- HDLC Tx/Rx 드라이버 처리기 프로세스 생성
- 명령어 해석기 호출

3.3.2 오류 처리기 (Error Handler)

오류 처리기는 리스트의 형태의 구조를 갖는 정보 집합체로 QUEST의 동작 중에 발생하는 각종 오류 기능을 진단하고 그에 맞는 오류 메시지를 사용자에게 제공한다. 오류 메시지 리스트는 부록을 참조한다. 오류 외에도 QUEST는 사용자에게 보다 정확한 상황 판단을 위한 정보의 제공을 위하여 동작 메시지와 주의 메시지 그리고 사용에 관한 메시지 등의 4개의 정보 메시지 군을 갖도록 설계한다. 모든 오류/주의/동작/명령어 사용 메시지들은 아래와 같은 고유한 구조를 갖는다.

```
Struct {
message-list : message-prefix _ command- list
message-prefix : E|W|A|U|S (E:error, W:warning, A:action,
U:usage, S:use)
command- list: command-primitive- list _ action_ list
command-primitive-
list: SHM|FILE|PRCS|LOG|TOG|SHELL|DGT|·
action-list:

OPEN|CREATE|NONE|FAIL|ERR|INV|OFF|ON|NULL|OVE
R|MORE|·
}
```

오류 메시지를 포함한 모든 정보 메시지는 특성별로 리스트화 하여 체계화하여 QUEST의 유지보수를 용이하게 한다. 오류 메시지 외의 다른 3개의 메시지에 관한 사항은 다음과 같다.

- 오류 메시지

오류 메시지는 시그널 및 메시지의 처리 불가, 그리고 사용자 쉘 명령어 이행 불가시에 사용자에게 제공되는 메시지이다. 일단 오류 메시지가 제공되면 사용자는 자신이 입력한 명령어 및 명령어에 관련된 정보를 정확하게 확인한 뒤 재시도해야 하며, 동작 중에 발생한 오류는 드라이버 인터페이스 보드를 포함한 시스템 및 송수신되는 시그널 또는 메시지 등을 확인한다.

- 주의 메시지

주의 메시지는 오류 메시지와 달리 기능은 정상적으로 수행되나 경미한 문제점 및 사용자에게 명령어 수행 환경에 따른 정보 제공 등을 위하여 출력되는 메시지이다. 이와 같이 주의 메시지가 필요한 것은 정보간의 AND/OR 형태로 존재하는 관계성을 명령어 처리기 및 쉘 명령어 프리미티브에서 검사하여 사용자에게 제공함으로써 사용자의 오류를 최소화하도록 하는데 있다.

- 동작 메시지

이 메시지는 오류나 주의 메시지와는 다른 특징을 갖는다. 동작 메시지는 시스템 동작 중에 어떤 원인에 의해서 결과가 발생할 때에 사용자의 주의를 환기시키기 위하여 제공하는 메시지이다.

- 명령어 사용 메시지

명령어 사용 메시지는 쉘 명령어 프리미티브를 쉘 환경에서 사용자가 입력한 명령어를 해석하여 명령어 프리미티브 풀에 있는 쉘 명령어인 경우에 한정하는 메시지로 쉘 명령어 프리미티브별 Synopsis에 맞지않는 명령어 정보를 포함하고 있으면 발생하는 메시지이다.

3.3.3 시그널 처리기 (Signal Handler)

사용자 응용 프로그램이나 다른 프로세서로부터 전송된 시그널을 추적(signal trace)하여 스크린에 출력하거나 사용자가 원하는 로그 파일에 출력하여 제공하는 기능이다. 이 시그널 처리기는 쉘 환경 처리기에서 사용자

로부터 입력된 명령어에 의해서 작동(invoke)되어 해당 시그널의 송수신 여부에 따라 시그널의 헤더나 메시지의 전체 부분을 제공한다. 또한 송수신 시그널에서 오류가 발생하면 시그널 오류 정보 파일에 저장하여 사용자의 쉘 명령어에 따라 시그널 오류 정보를 제공하여야 한다. 시그널 처리기의 주요 기능은 다음과 같다.

- 시그널 추적 기능 활성화 및 비활성화
- toggle 및 logging 기능 수행
- 시그널 오류 검사
- 시그널 추적 정보 제어 및 감시
- 시그널 전송 속도 측정
- 시그널 자동 전송
- 시나리오에 의한 시그널 입출력 모니터링

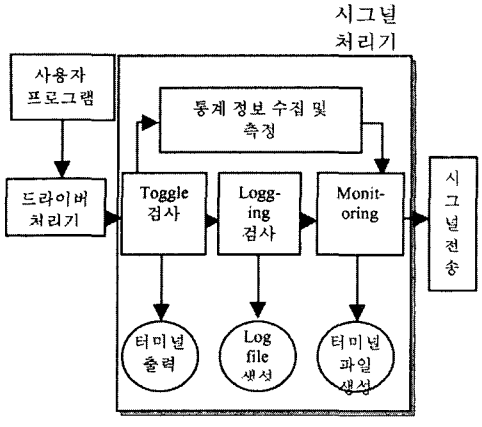

그림 3 시그널 처리 과정

가) 입출력 시그널 추적 기능

입출력 시그널 추적 기능은 송수신된 프로세스 또는 시그널과 입출력 시그널 추적 정보에 등록된 프로세스 또는 시그널과 일치하면 toggle과 logging 여부에 따라 표준 출력 터미널 또는 사용자에 의해 지정된 로그 파일 상에 시그널 및 메시지 정보를 저장하는 기능이다. 입출력 시그널은 위에서도 언급한 바와 같이 시그널 추적 기능은 대상에 따라 프로세스 레벨과 프로세스에 속한 시그널 레벨 등의 두 가지로 나누어 진다. 프로세스 레벨은 프로세스에 대한 모든 입출력 시그널을 대상으로 하여 추적 기능이 활성화되는 것이며, 시그널 레벨은 프로세스가 아닌 특정 시그널을 대상으로 해서 추적 기능이 활성화 되는 것이다. 다음의 그림은 프로세스와 시그널과의 관계를 시그널 추적 기능 측면에서 본 것이다.

- 프로세스 레벨

입출력 되는 시그널을 프로세스 ID 정보에 의해 해당 시그널 및 메시지를 추적한다. 따라서 프로세스에 속한 모든 시그널들을 대상으로 한다.

- 시그널 레벨

프로세스 레벨이 프로세스에 속한 모든 시그널을 대상으로 하는 것에 비해 시그널 레벨은 지정된 시그널 ID 정보에 의해 해당 시그널 및 메시지를 추적한다.

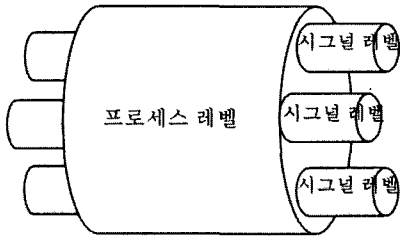

그림 4 시그널 제어 레벨

나) 시그널 추적 정보 예약 및 감시

QUEST는 사용자가 지정한 복수개의 시그널에 대한 추적 정보를 예약하거나 등록된 시그널을 활성화 또는 비활성화 시키고 시그널의 활성화 및 비활성화를 위한 데몬 프로세스를 이용하여 시그널 추적 정보를 감시한다. 복수개의 등록된 시그널 정보 중에 FIFO(first-in first-out) 원칙에 의하여 먼저 등록된 시그널로부터 활성화 시킨다. 활성화 정보에는 점화 시간(trigger time)과 지속 시간(duration time) 이 있어 점화 시간이 0이 되면 지속 시간 동안 활성화 되는 메커니즘을 갖는다. 따라서 지속 시간이 경과하면(expire), **idle** 상태로 천이한다. 또한, 본 기능을 구현하기 위하여 초기 상태(init), 활성화 상태(act), 비활성화 상태(deact), 그리고 휴지 상태(idle)로 구분하여 각 상태에 따라 시그널 추적 기

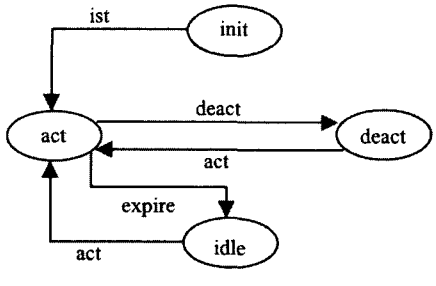

그림 5 시그널 추적 예약 상태 천이도

능이 동작하게 하였다. 다음 그림은 각 상태별 천이도를 나타내고 있다.

3.3.4 명령어 해석기 (Command Interpreter)

명령어 해석기는 쉘 환경 처리기의 중요한 부분으로서 사용자의 명령어를 받아 문법(syntax check)을 검사하여 오류가 없으면 파싱(parsing)을 통하여 토큰(token)을 생성한다. 생성된 토큰은 명령어 처리기로 보내져 관련 명령어를 활성화시킨다. 명령어 해석기의 기능은 다음과 같다.

- 명령어 수락
- 명령어 문법 검사
- 명령어 라인 파싱 및 토큰 생성
- 명령어 처리기와 연동

3.3.5 명령어 처리기 (Command Handler)

명령어 해석기로부터 제공된 토큰을 확인하여 명령어 프리미티브 풀(command primitive pool)을 조사하여 해당 명령어가 존재하는 지를 확인하여 존재하면 명령어를 토큰과 함께 활성화시킨다. 명령어가 명령어 프리미티브 풀에 존재하지 않으면 QNX 시스템 라이브러리 function을 호출한다. 명령어 해석기의 기능은 다음과 같다.

- 명령어 프리미티브 풀 검사
- 명령어 권한 검사(non-authorized command filtering)
- 명령어 활성화
- 명령어 예약
- Batch 파일에 의한 명령어 일괄 수행

3.3.6 부팅 처리기 (Booting Handler)

쉘 명령어나 시스템 로더에 의해 사용자 프로그램을 다시 부팅하기 위하여 데몬(daemon) 형태로 감시 프로세스를 제공한다. 부팅 처리기에서는 쉘 명령어에 의한 부팅 요구가 들어오면 Rx/Tx 드라이버 처리기와 시스템 로더를 재 실행하고, 시스템 로더로부터 부팅 요구가 들어오면 QUEST 프로세스 테이블에 있는 사용자 프로세스들을 모두 제거한 뒤 데이타베이스를 참조하여 사용자 프로세스 테이블을 재구성한다.

- 부팅 요구 접수
- 드라이버 처리기와 시스템 로더의 재실행
- QUEST 프로세스 테이블의 재구성

3.4 명령어 프리미티브 풀 (Command Primitive Pool)

3.4.1 개요

QUEST 시스템과 사용자를 분리시켜 QUEST의 독립성을 높임으로써 명령어의 추가와 삭제가 매우 용이하게 하여야 한다. 따라서 명령어 프리미티브 풀은 사용자로 하여금 보다 쉽게 QUEST에 접근할 수 있는 친숙성과 시스템의 안정성을 향상시킬 수 있도록 설계되어야 한다. 이 풀은 공유 메모리(shared memory)를 통하여 QUEST에 빠르게 접근할 수 있으며, 풀에 있는 명령어 프리미티브들 또한 각각이 서로 독립적으로 활성화되게 된다. 명령어가 예약 및 활성화되는 과정은 다음 그림과 같다.

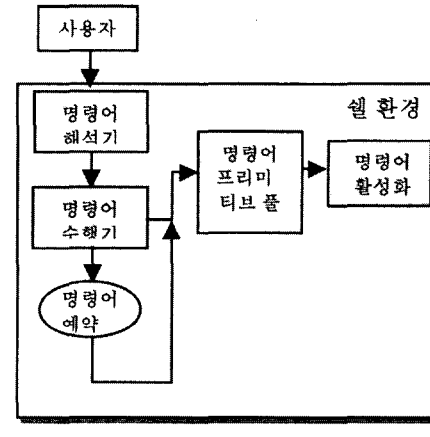

그림 6 명령어 예약 활성화 과정

3.4.2 프로세스 처리기 (Process Manage)

사용자는 QUEST상에서 다양한 프로세스 처리 기능을 갖는다. 프로세스의 실행(run), 프로세스의 소멸(kill), 프로세스 교체(swap) 등의 쉘 명령어 프리미티브를 이용하여 QUSET 프로세스 상태 테이블과 연동해서 프로세스를 처리한다.

3.4.3 파일 처리기 (File Manage)

사용자는 QUEST상에서 다양한 파일 처리 기능을 갖는다. 파일의 추가, 삭제 등을 쉘 명령어 프리미티브를 이용하여 QUSET 프로세스 상태 테이블과 연동해서 처리한다.

3.4.4 사용자 접근 권한 제어 처리기 (User Access Authority Control Manage)

각 사용자는 접근 권한 제어 기능을 통하여 쉘 명령어를 사용할 수 있으며, 명령어 사용에 대한 정보가 로

그 파일에 저장된다. 사용자 접근 권한을 부여 받지 못한 경우에는 쉘 명령어의 일부를 접근할 수 없게 된다. 또한 사용자 접근 권한을 부여 받기 위해서는 사용자 로그인에 따른 패스워드를 제시하여 접근할 수 있도록 설계한다.

이 외에도 시그널 추적 처리기, 시그널 자동 전송 처리기, Logging 처리기, Toggle 처리기, 이력 제어 처리기, 형상 제어 처리기, 측정 및 통계 기능 처리기, Boot 처리기, 도움말 처리기 등이 있다.

3.4.5 측정 및 통계 처리기 (Measurement and Statistics Manage)

쉘 명령어 이용하여 지속 시간이 주어 졌을 경우에 입력 시그널에 대한 측정 시간 동안의 도착 수, 도착률, 트래픽 밀도 등에 관한 측정 정보를 얻을 수 있다. 또한, 통계 명령어를 이용하여 시그널, 프로세스, 파일, 명령어 등에 관한 저장된 통계 데이타를 수집하거나 검색할 수 있어 시스템의 사용 상태나 현재의 시스템 상태를 분석할 수 있다.

4. 성능 분석

4.1 시뮬레이션 환경

QUEST의 지연 시간 및 전송 속도에 관한 성능을 분석하기 위하여 공유 메모리에 대한 접근 상태에 따라 다음과 같은 시뮬레이션 환경을 구성하였다 [9]. 물론 QUEST 시스템이 호제어 프로세서의 성능에 영향을 미치는 요인으로 큐 상태, 전송 속도, 프로세스 동작 상태, 부하 상태 등이 있으나 본 논문에서는 프로세스간 통신 측면에서 메시지의 지연 시간을 변화하면서 전송 속도를 측정하였다. 전송 속도를 성능 분석의 요소로 선택한 것은 본 논문에서 제안하고 있는 QUEST 시스템이 호제어 프로세서에 실장되는 호처리 및 유지보수 프로그램과 함께 실장되어 동작하는 시험이나 디버깅용의 지원 시스템이므로 QUEST 시스템이 운용 중에 호처리 및 유지보수 프로그램에 영향을 주어서는 안되기 때문에 QUEST 시스템을 동작하는 상태하에서 응용 프로그램에 미치는 영향을 분석하였다. 응용 프로그램은 FSM (finite state machine) 원리에 따라 메시지를 다른 프로세서나 기지국 제어 프로세서내의 다른 프로세스와 통신한다. 통신의 지연 및 전송 속도에 따라 응용 프로그램의 실행에 많은 영향을 끼치게 된다. 따라서 본 논문에서는 QUEST가 동작하고 있을 때 메시지의 지연 시간에 따라 전송 속도가 어떻게 변하는 가를 살펴 보았다. 또한 QUEST 시스템의 동작이 응용 프로그램에 미치는 실질적인 영향을 분석하기 위해서 해석적 접근 방법보다 시뮬레이션을 통하여 실시하였다.

- QUEST의 실행 본체는 PC 166MHz을 사용
- HDLC 메시지 전송을 위하여 PT334 보드의 송수신 단자에 루프백 설치
- 사용한 공유 메모리는 약 4x4Kbytes로 Toggle시 정보를 얻기 위하여 사용
- 시그널의 규칙적인 발생을 위하여 배치 파일 처리 명령어 사용
- 시그널의 발생을 위하여 특정 시그널 송수신을 위한 프로세스 생성, Toggle off/on, 시그널 자동 전송 명령어 이용

4.2 가정 사항

시뮬레이션을 위한 가정 사항은 다음과 같다.

- 한번 전송 시 송수신하는 메시지는 100bytes로 한다.
- 1회부터 100회까지 10회씩 증가하면서 반복하여 전송할 수 있도록 한다.
- 매 송신마다 약 0부터 10msec까지 전송 지연 시간을 사용하여 실제의 전송 시 생기는 전송 지연 효과로 가정한다.
- 출력시의 성능 분석을 위하여 매 5초 마다 특정 시그널을 전송하고 시그널의 헤더 부분을 표준 입출력 터미널을 통하여 출력함으로써 출력에 소용되는 시간을 고려한 전송 속도를 분석한다.
- 성능 분석에 사용된 시그널의 크기는 234Bytes로 고정한다.

4.3 결과 분석

4.3.1 출력이 없는 상태하에서의 Toggle 상태에 따른 전송 속도 분석

그림 7은 전송된 메시지가 표준 입출력 터미널 상에 출력하는 상태 (Toggle = off)에서 지연 시간의 변화에 따라 전송 속도를 측정하였다. 반면에 그림 8은 전송된 메시지가 표준 입출력 터미널 상에 출력되지 않도록 하는 상태 (Toggle = on)에서 지연 시간의 변화에 따라 전송 속도를 측정하였다.

성능 분석 결과 지연 시간이 0msec일 경우에 약 500Kbps의 전송 속도를 가지나 지연 시간이 10.0msec 이상일 경우에는 약 45Kbps의 속도를 유지하는 것으로 밝혀졌다. 지연 시간이 1.0msec 이상에서는 지연 시간에 상관 없이 전송 속도가 일정하므로 HDLC 드라이버

와 QUEST가 QNX 시스템에 영향을 주지 않고, 안정적으로 동작함을 알 수 있다. 특히 사용자는 메시지를 전송 할 때에 지연 시간이 없도록 프로그래밍을 해야 전송 속도를 높일 수 있다. 그리고 Toggle 상태가 on인 경우에는 off 인 경우에 비해 크게 차이가 없으나 지연 시간이 0msec일 경우에는 약 200Kbps의 속도 저하가 발생하는 것을 보이고 있다. 이 것은 Toggle을 위하여 공유 메모리를 접근하는데 소요되는 시간이다. 그러나 지연 시간이 0msec보다 큰 경우에는 공유 메모리의 접근 시간보다 지연 시간이 크므로 공유 메모리의 접근 시간이 상대적으로 큰 영향을 끼치지 않는다.

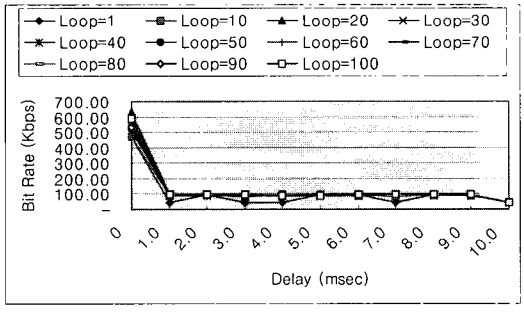

그림 7 전송 횟수(Loop) 변화에 따른 지연시간 대 전송속도 성능분석 (Toggle=off)

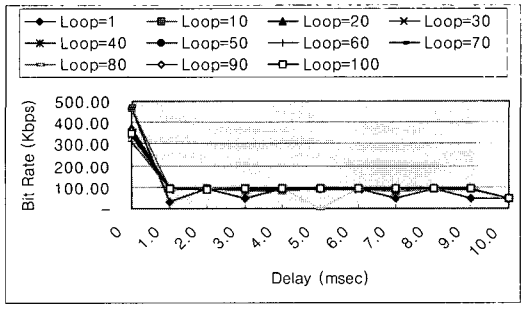

그림 8 전송 횟수(Loop) 변화에 따른 지연시간 대 전송속도 성능분석 (Toggle=on)

4.3.2 출력하는 상태하에서의 전송 속도 분석

그림 9는 Toggle이 on 상태에서 5초 간격으로 약 234 Bytes의 데이타 크기를 갖는 메시지를 송수신(루프백) 하면서 출력단 (HDLC 드라이버 송신 인터페이스 측)에서의 헤더 메시지를 표준 입출력 터미널상에 출력할 경우의 전송 속도를 지연 시간 및 전송 횟수의 변화에 따라 성능 분석을 실시하였다.

성능 분석 결과에 따르면 지연 시간이 0msec인 경우, 그림 8에 비해 약 25%의 성능 저하가 생겼으나 1.0msec 이상에서는 동일한 전송 속도를 갖는 것으로 분석되었다. 이러한 결과는 지연 변화에 따른 QUEST의 성능 변화는 없는 것을 보여 주고 있다. 또한, 전송 반복 횟수가 증가해도 93.76Kbps의 속도를 유지하고 있음을 보여 주고 있다. 즉, 반복 횟수가 증가하게 되면 시스템에 의해 받는 영향이 줄어들어 전송 속도에 큰 영향을 미치지 않게 된다.

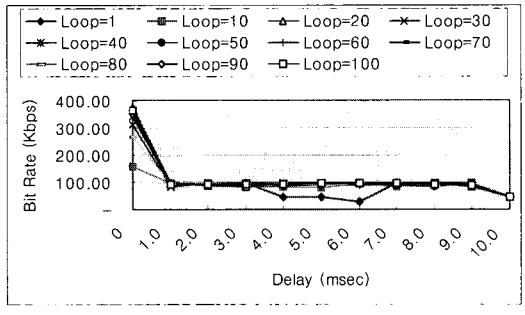

그림 9 전송 횟수(Loop) 변화에 따른 지연시간 대 전송속도 성능분석

4.3.3 시그널 헤더 출력 상태에 따른 전송 속도 분석

그림 10은 시그널 헤더 출력 상태를 on과 off 상태하에서 지연 시간이 0msec일 때에 전송 횟수를 변화해 가면서 전송 속도를 비교 분석하였다. 분석 결과 전송 횟수가 10 이상에서는 시그널 헤더 출력 상태와 상관없이 동일한 전송 속도를 보이나 10보다 적은 경우에는

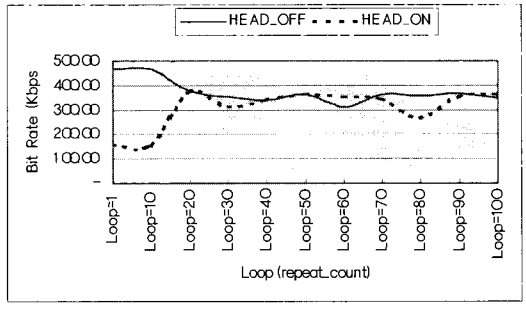

그림 10 시그널 헤더 출력 상태에 따른 횟수(Loop) 변화 대 전송속도 성능분석

약 300Kbps의 속도차를 보인다. 이러한 결과는 지연 시간을 고려하지 않고 시그널이 반복적으로 계속해서 발

생하게 되면 일정 수준 이하에서는 표준 입출력 터미널에 시그널의 헤더 부분이 출력되는 시간으로 인하여 성능 저하가 생긴다. 그러나 이러한 현상도 시그널 헤더 출력이 반복 횟수가 10 이상에서는 두 경우 모두 350Kbps 속도를 유지하고 있음을 보여 주고 있다. 즉 반복되는 시그널의 전송시 특정 시그널의 표준 입출력 터미널 상의 출력이 QUEST의 기능에 큰 영향을 미치지 않고 안정적으로 동작하고 있음을 확인할 수 있다.

5. 결 론

본 논문에서는 QNX에 기반한 IMT-2000 기지국 제어기내의 호제어 프로세서의 사용자 실시간 디버깅을 위한 지원 도구로서 QUEST라는 QNX 기반의 실시간 사용자 환경 지원 도구를 설계 및 구현하였다. QUEST는 쉘 환경을 통하여 시그널 제어, 프로세스 제어, 파일 제어, 사용자 접근 권한 제어, 명령어 제어 등의 다양한 기능을 갖추고 있다. 또한 TCP, HDLC, 그리고 ATM과 같은 여러 가지 매체들을 제어하는 드라이버 제어기를 내부적으로 관리함으로써 사용자와의 완벽한 분리를 통하여 매체에 상관없이 데이타를 주고 받을 수 있게 되었다. 마지막으로 메시지의 지연 시간을 변화해 가면서 전송 속도에 대한 성능을 분석한 결과, 표준 입출력 터미널에 메시지를 출력하면서 메시지의 전송 횟수를 증가했을 때에도 메시지의 전송 속도에 급격한 성능 저하가 일어나지 않아 QUEST가 응용 프로그램에 미치는 영향이 크지 않음을 알 수 있다. 즉, QUEST 시스템이 호처리나 유지보수와 같은 응용 프로그램의 성능을 저하시키지 않음을 확인하였다.

본 QUEST는 이동통신 시스템 뿐만 아니라 분산 환경에서 TCP, HDLC, 그리고 ATM 드라이버를 통한 시그널 송수신 시에 사용자에게 편리하게 사용할 수 있으며, 쉘 명령어의 수는 약 50여개이고, 전체 라인 수는 약 2만 여 라인으로 Watcom C로 구현하였다. 현재는 TCP 및 HDLC 드라이버 처리기에 대해 시험을 완료했으며, 앞으로 ATM 드라이버 처리기를 추가하여 시험을 실시할 예정이다.

참 고 문 헌

[1] CROS-Concurrent Real-time Operating System, ETRI, 1994.

[2] ATM용 CROS-Concurrent Real-time Operating System, ETRI, 1995.

[3] Y. N. Han, K. C. Han, and H. G. Bahk, "A CDMA-based Digital Cellular Infrastructure: CDMA Mobile System(CMS)," *International Workshop on Multi-dimensional Mobile Communications*, 1994.

[4] J. H. Rhee, S. J. Lee, S. Y. Lim, and W. G. Park, A Highly Reliable Maintenance and Administration Software Architecture in the Base Station for IMT-2000, *Proceedings of the 4th International Workshop on Mobile Multimedia Communications (MoMuC97)*, pp.421-424, Seoul, September 1997.

[5] 이제헌, 이숙진, 박우구, QUEST-IMT-2000 표준 모델 기지국 제어 소프트웨어 개발 환경 지원 도구, 제 3회 통신 소프트웨어 학술대회, pp.307-310, 1998.

[6] QNX Software System Ltd, QNX Operating System : System Architecture, 1996.

[7] QNX Software System Ltd, Watcom C Library A to M, 1996.

[8] QNX Software System Ltd, Watcom C Library N to Z, 1996.

[9] J. Banks, J. S. Carson II, and B. L. Nelson, Discrete-Event System Simulation, Prentice Hall International Editions, 1996.

박 우 구

1983년 2월 홍익대학교 전자계산학과졸업(이학사). 1993년 2월 한남대학교 전자계산공학과 졸업(공학 석사). 1999년 2월 충북대학교 전자계산학과 졸업(이학 박사). 1983년 3월 ~ 현재 한국전자통신연구원 무선방송기술연구소 동기시스템기술연구실 책임연구원. 정보처리 기술사. 관심분야는 CDMA 이동통신, 무선멀티미디어, 트래픽 제어, 분산 시스템, 시뮬레이션, 시스템 프로그래밍

이 제 헌

1993년 2월 숭실대학교 전자계산학과 졸업(공학사). 1995년 2월 숭실대학교 전자계산학과 졸업(공학 석사). 1995년 ~ 현재 한국전자통신연구원 무선방송기술연구소 통화제어연구실 연구원. 관심분야는 무선 ATM, 무선 시스템, 프로토콜 공학