TMN을 위한 실제 자원 시뮬레이터 설계

정회원 송 병 권*, 김 건 웅**, 진 명 숙***

Design of a Real Resource Simulator for TMN

Byung-kwen Song*, Geonung Kim**, Myung-sook Jin*** *Regular Members*

요 약

본 논문에서는 실제 자원의 개발 전에도 망 관리 시스템의 개발 및 운용 테스트를 수행하도록 지원하는 실제 자원 시뮬레이터(RRS: Real Resource Simulator)를 제안한다. RRS는 객체의 상태를 유지하는 MOT(Managed Object Table)와 사용자가 정의한 동작 특성을 유지하는 SDT(Simulation Data Table), 랜덤(Random) 값과 랜덤 주기(interval) 값을 발생할 지원 함수들, 순차적인 사건 발생 또는 값의 수정을 지원하는 스케줄링 테이블, 그리고 이들을 전체적으로 관장하는 메인 커널로 이루어져 있다. 본 논문에서는 이러한 각 요소의 기능과 동작 시나리오, 이를 이용한 평가 방안을 소개한다.

ABSTRACT

In this paper, we propose a RRS(real resource simulator) that supports the development and operational test of network management system before the development of real resources. The components of the RRS are the MOT(Managed Object Table) that holds the current status information of real resources, the SDT(Simulation Data Table) that holds the characteristics of real resources defined by user, the support functions that generate the random values and random interval values, the scheduling table that holds the sequence of events, and the main kernel. We describe the activities of each components and the operational scenarios.

Ⅰ. 서 론

정보화 사회의 기반인 통신망은 사회적인 요구에 따라 PSTN, PSDN, B-ISDN 등의 유선망과 CDMA를 기반으로 한 무선망이 서로 혼합된 형태로 복잡하게 진화되고 있다. 이에 따라 통신 서비스도 기존의 음성에서 데이터, 화상, 이동 컴퓨팅 서비스 등으로 점점 고도화, 다양화되고 있는 실정이다. 이러한 통신망 환경에 대응하여 ITU-T는 개별 운용시스템의 한계성을 극복하고 표준화된 개방형 방식의 총체적이고 일원화된 통신망 운용관리체제를 구축하기 위하여 TMN(Telecommunication Management Network) 관리 개념을 권고했다. TMN은 관리 정보 모델링을 위해 OSI(Open System Interconnection)에서 제안한 객체지향 모델링 기법을 채택하고 있고, 관리 정보의 접근 및 교환을 위하여 OSI의 CMIP(Common Management Information Protocol)/CMIS (Common Management Information Service) 프로토콜을 채택하고 있다[1][2][3][4][5].

현재 ATM(Asynchronous Transfer Mode) 기반 초고속 정보통신망, 지능망(IN: Intelligent Network), 정보망(TINA: Telecommunication Information Network Architecture)에서도 관리를 위한 기반 플랫폼으로 TMN을 채택하고 있고, PCS(Personal Communication System) 및 IMT-2000 등의 이동통신망에서도 기존 유선망과의 통합 관리를 위하여 TMN을 사용하고 있는 실정이다.

TMN은 그림 1과 같이 망 관리 역할에 따라 관

* 서경대학교 정보통신공학과 (bksong@skuniv.ac.kr),
** 목포해양대학교 해양전자통신공학부(kgu@mail.mmu.ac.kr),
*** 명지전문대학 정보통신과 (msjin@mail.mkc.ac.kr)
논문번호 : K01031-0118, 접수일자 : 2001년 1월 18일
※ 본 연구는 목포해양대학교 교내연구비 지원으로 일부 수행되었습니다.

리자(manager)와 에이전트(agent)로 구분된다. 관리자는 에이전트에게 관리 요청을 전달하고 에이전트로부터 관리 요청에 대한 수행 결과를 반환 받거나, 특별한 사건이 발생했음을 알리는 통고(notification) 메시지를 수신한다. 또한 에이전트는 관리자로부터 관리요청을 수신한 다음 실제 자원(real resource)에 접근하여 해당 값을 가져온 후, 그것을 관리자에게 반환하거나 실제 자원에서 발생한 통고를 전달한다. 따라서 관리자는 에이전트를 통하여 실제 자원에 대한 관리를 종합적으로 수행한다.

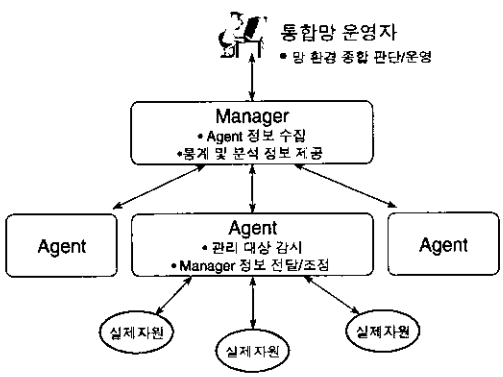


그림 1. TMN 관리 모델

망 관리 시스템에서 실제 자원은 전기통신망을 구성하는 교환기 및 각종 유무선 통신 장비 등 실제 관리하고자 하는 하드웨어 또는 소프트웨어적인 요소들이다[6][7][8][9][10]. 이러한 실제 자원을 관리하기 위한 망 관리 시스템은 실제 자원이 완성된 후에 개발을 시작하거나, 실제 자원 개발자로부터 사전에 관리 정보를 제공받아 실제 자원 개발과 병행해서 개발한다. 그러나 전자의 경우는 완성된 실제 자원을 기존 통신망에 설치 및 운용했을 때, 해당 장비에 대한 관리 시스템 부재로 일정 기간 동안 관리 공백이 초래되고, 후자 또한 실제 자원 개발자로부터 얻는 정보의 부족으로 인해, 양쪽이 완성된 이후에도 일정 기간 동안 정합시험이나 안정성 시험을 거치면서 문제점을 수정 및 보완해야 한다. 또한 망 관리 시스템이나 에이전트들이 개발된 후에 망 관리 시스템 자체에 대한 기능 및 성능을 분석하고자 하면, 실제 망 요소들을 이용해 테스트 환경을 갖춘 후 이를 수행해야 하는데, 이러한 환경 구축에 소요되는 비용이나 시간도 많이 필요하다.

이러한 문제점들을 해결하기 위해선, 실제 자원 개발 전에도, 사용자가 지정하는 형태로 속성(attribute) 값들이나 사건(event)을 생성하여 에이전트에게 돌려주고, 또한 실제 자원 없이도 운용 환경을 꾸밀 수 있도록 하는, 실제 자원 시뮬레이터가 필요하다. 본 논문에서는 이를 RRS(Real Resource Simulator)로 지칭한다. RRS의 주 역할은 망 관리 시스템이 관심을 갖는, 실제 자원이 동작하면서 발생하는 속성 값들의 변화나 사건 등을 사용자가 원하는 방식으로 생성시켜, 관리시스템의 개발 및 운용 테스트를 하도록 하는 것이다. 다음 그림 2는 관리자, 에이전트 그리고 본 논문에서 제안하는 RRS와의 관계를 나타낸다.

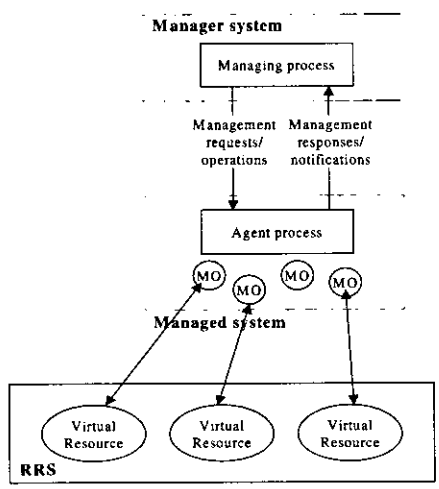


그림 2. 관리자, 에이전트, 그리고 RRS와의 관계

현재 TMN에 관련된 연구는 이미 개발된 플랫폼(platform)을 이용하거나 독자 개발된 것을 사용하여 특정 통신망 환경에서 구동되는 관리 시스템의 기능이나 구현 메커니즘에 관한 것이 대부분이며[11][12][13][14], 개발이 완료된 망 관리 시스템의 안정성이나 동작을 검증하고 또한 가상의 실제 자원을 이용하여 망 관리 시스템을 조기 개발하는 분야에 대해서는 연구결과를 찾기 어렵다.

본 논문에서는 먼저 II장에서 RRS의 전체 구성과 각 요소들에 대해 기술하고, III장에서 RRS의 각 동작 시나리오를 소개한다. IV장에서는 RRS를 지원하기 위한 확장된 GDMO 문법과 GDMO 컴파일러에 대하여 소개하고, V장에서는 RRS를 이용한 망관리 에이전트 및 관리시스템의 성능 평가 방안에 대해 언급한 후 VI장에서 결론을 맺는다.

II. RRS의 전체 구성

RRS는 그림 3과 같이, 에이전트와의 다양한 통

신 방식을 지원할 통신 모듈, 각 관리 객체 별로 사용자가 지정한 속성 값 및 통고의 생성 방식 등을 담고 있는 SDT(Simulation Data Table), 현재 생성된 관리 객체들의 정보를 담고 있는 MOT(Managed Object Table), 이들을 바탕으로 속성 값을 변화시키거나 통고를 발생시킬 커널 메인(kernel main)과 이때 이용할 지원 함수(support function)코드와 스케줄링 테이블, SDT 또는 객체 테이블 내의 값 변경, 통고의 발생 등을 직접 수행할 수 있도록 하는 GUI 인터페이스로 이루어져 있다.

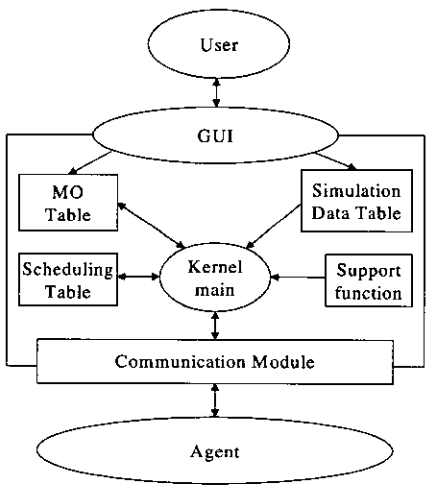


그림 3. RRS 전체 시스템 구조

1. 커널 메인

여기서 커널 메인은 ① 에이전트로부터 요청 받은 속성 값을 MOT에서 찾아 반환하고, ② 사용자가 지정한 자원함수에 따라 사건을 발생시키고, 이에 대한 통고 메시지를 생성, 에이전트에게 전달하는 기능을 수행한다. 따라서 커널 메인은 RRS내의 모든 자원들을 관리하게 되는데, 이에 관련된 관리 객체의 정보들이 MOT에 존재하게 되고, 각 자원마다 사용자가 지정한 시뮬레이션 동작 특성이 SDT에 존재하게 된다. 또한 순서적으로 사건을 발생시키기 위하여, 사건이 발생할 시간에 따라 등록하게 되는 스케줄링 테이블과 운영체제의 타이머를 이용한다.

2. 통신 모듈

실제 자원과 에이전트간의 통신은 다양한 프로토콜 스택이 이용되고 있다. 보통 이더넷(Ethernet)을 기반으로 하여 IP(Internet Protocol), TCP(Transmission Control Protocol), UDP(User Datagram Protocol)를 이용하는 방식이 대부분이지만, 직렬(serial line) 연결 방식도 이용하기도 하고, 또한 ATM, Frame Relay 등 다양한 전송 방식을 이용할 수도 있다. 우선은 가장 일반적인 TCP 또는 UDP 연결을 이용하도록 하며, 추후 각 통신 모듈을 추가하도록 한다.

3. 지원 함수

여기에는 커널 메인에서 이용할 랜덤 값을 생성하는 함수들이 존재한다. 이러한 지원함수는 크게 두가지 용도로 쓰이게 되는데, 상태 값 자체의 생성과 사건이 일어나는 시간 간격 결정에 이용된다. 그림 4는 RRS에서 지원할 함수들이다.

Double expo(double a):
Random Number 분포:Exponentially

$$f(x)=\begin{cases}\lambda e^{-\lambda x} & x \geq 0\\ 0 & x < 0\end{cases}$$

Parameter: a – means(λ)

Double unifc(double a, double b);
Random Number 분포:Continuous Uniform

$$f(x)=\begin{cases}\dfrac{1}{b-a} & a < x < b\\ 0 & \text{otherwise}\end{cases}$$

Parameter: a – low bound, b – high bound

long unifd(int i, int j);
Random Number 분포:Discrete Uniform

$$f(x)=\begin{cases}\dfrac{1}{b-a} & a < x < b\\ 0 & \text{otherwise}\end{cases}$$

Parameter: I – low bound(a), j – high bound(b)

long bino(int n, double p);
Random Number 분포:Binomial

$$p(i)=\binom{n}{i}p^{i}(1-p)^{n-i}$$

Parameter: n – number of trial(n), p – probability

long poisson(double a);
Random Number 분포:Poisson

$$p(i)=e^{-\lambda}\frac{\lambda^{i}}{i!} \qquad (i \geq 0)$$

Parameter: a – means(λ)

Double gaussian(double a, double d);
Random Number 분포:gaussian(Normal)

$$f(x)=\frac{1}{\sqrt{2\pi\delta}}e^{-(x-\mu)^2/2\delta^2} \qquad -\infty < x < \infty$$

Parameter: a – means(μ), b – variance(δ)

그림 4. 지원 함수

4. 스케줄링 테이블

각 관리객체 인스턴스마다 다음 사건이 일어날 시간 간격을 결정하는데 이용하는 지원 함수와 매개변수(parameter)들이 다르기 때문에, 나중에 수행하여 결정된 사건의 발생이 먼저 결정된 사건보다

선행되어야 하는 경우도 생긴다. 따라서, 이를 위하여 지원 함수의 결과에 따른 사건 발생 시간을 기준으로, 순차적으로 정렬된 테이블이 필요하다. 이러한 테이블의 각 열에서는 관리객체 식별자와 상대적으로 계산된 시간 값을 가지고 있다. 시간 값의 단위는 ms로 하며, 실제로 원활한 값의 삽입을 위해서, 연결된 리스트(linked-list) 구조를 갖도록 한다. 새로 사건 발생 시간이 결정되면, 이에 따라 해당 위치에 삽입해야 하는데, 다음 그림 5는 (a)상태에서 새로 200ms후에 발생할 sample9 사건을 삽입한 결과를 (b)에서 보이고 있다.

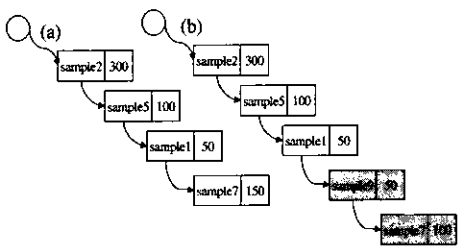


그림 5. 스케줄링 테이블 예

실제로 처음에 있는 사건 시간부터 고려하여 삽입해야 하지만, 이미 이에 대한 시스템 시간은 진행중이므로, 시간이 상대적으로 작은 경우에는 두 번째 있는 사건 시간부터 고려하였다. 여기서 보면 두 번째 사건부터 고려하여, sample1과 sample7 사이에 삽입되면서, 자신의 대기 시간은 전체 200에서 앞선 두 사건들의 시간의 합 150을 감한 50이 되고, 다음 사건 sample7은 원래 시간에서 방금 삽입된 사건으로 인한 시간 50이 감해져, 150에서 100으로 줄어드는 것을 볼 수 있다.

5. SDT

SDT에는 각 관리 객체의 속성 값 및 통고들을 생성할 때 이용할 정보들을 담고 있다. 관리 객체마다 여러 속성과 사건 발생이 가능하므로, 관리 객체 식별자와 속성 식별자, 사건 식별자를 이용, 구분하도록 하고, 랜덤 값을 생성하는데 이용할 지원함수, 사건 발생 간격을 결정할 지원 함수, 이들 지원 함수들의 관련 매개 변수들을 담고 있다. 그림 6은 SDT의 예를 보여주고 있다.

MO id	AttrId /EventId	syntax	init value	gen fn	interval fn	param1	...	paramN
sample1	Asample1	char	ABC					
sample2	Asample2	integer	4522346	uniform				
sample3	Asample3	real	3.132					
sample4	Esample1				poisson	100.0		

그림 6. SDT 예

6. MOT

MOT에는 실제 자원의 현재 값들이 저장된다. 실제 자원 상태는 바로 전 상태에 종속적인 것과 독립적인 것이 있을 수 있는데, 종속적인 경우 바로 전 상태 값을 바탕으로 새로운 상태 값이 생성되어야 한다. 따라서 이를 위해 현재 상태 값들을 저장할 MOT가 있다. 그림 7은 MOT의 예를 보여주고 있다. 여기서 정수나 실수인 경우 바로 값을 저장하고, 문자열인 경우 주소 값을 저장한다.

MO id	AttrId	syntax	value
sample1	Asample1	char	ABC
sample2	Asample2	integer	4522346
sample3	Asample3	real	3.132
sample7	Asample2	integer	4522346

그림 7. MOT의 예

7. GUI

GUI 인터페이스는 커널 동작 전에 SDT의 값들을 초기화하거나, 동작 중에도 사용자의 뜻에 따라 MOT에 저장된 정보들을 변경할 수 있도록 한다. 그림 8은 SDT에 저장된 속성의 지원 함수 종류, 파라미터, 그리고 주기에 관한 정보를 설정하는 예다. 이러한 동작은 커널의 실행 전 또는 실행 중에도 이루어질 수 있다.

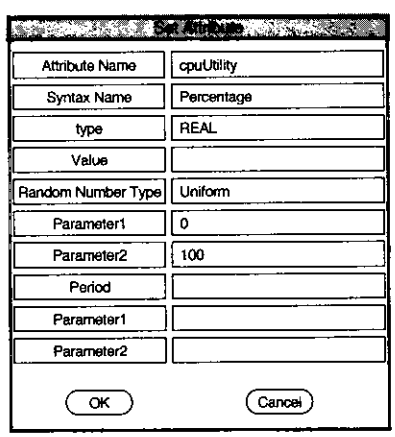


그림 8. SDT 정보 설정 화면의 예

Ⅲ. RRS의 동작 시나리오

1. GUI를 통한 SDT 정보 설정

그림 9는 사용자가 GUI를 통하여 SDT에 있는 값을 초기화시키고, 그 결과를 바탕으로 RRS가 MOT와 스케줄링 테이블을 초기화시키는 과정을 보여준다.

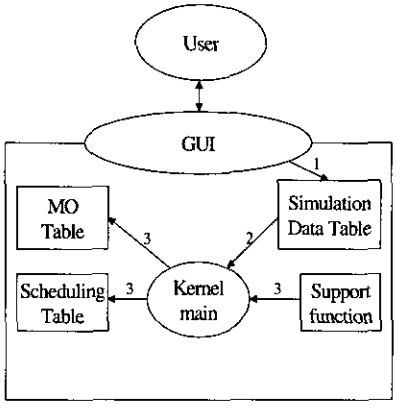


그림 9. GUI를 통한 SDT 초기화 및 동작

① 사용자는 GUI를 통하여 특정 관리 객체의 초기 값, 랜덤한 속성 값을 생성하기 위한 지원 함수, 사건 또는 값의 변경이 일어날 시간 간격을 결정할 지원 함수, 그리고 각 함수들의 매개 변수들을 설정한다.

② RRS가 기동하게 되면, 커널은 사용자가 설정한 값들을 SDT로부터 읽어온다.

③ 커널 메인은 SDT에서 읽어온 값을 이용하여 MOT를 초기화하고, 다음 사건 또는 속성 값의 수정이 이루어질 시간을 결정하여 스케줄링 테이블에 등록한다.

RRS 동작 중에도 사용자는 GUI를 통하여 SDT에 저장된 값들을 변경할 수 있는데, 이 경우 해당된 MO들에 대해서만 값을 수정하고, 다음 사건 또는 변경이 일어날 시간을 결정하여 스케줄링 테이블에 등록한다.

① 사용자는 GUI를 통하여 특정 관리 객체의 값, 랜덤 생성 지원 함수 및 랜덤 간격 결정 지원 함수, 그리고 각 함수들의 매개 변수들을 수정한다.

② 변경된 경우, 이를 인터럽트 등의 통신 수단을 통해 커널 메인에게 알린다.

③ 커널 메인은 SDT에서 변경된 값을 이용하여 MOT를 수정하고, 다음 사건 또는 속성 값의 수정이 이루어질 시간을 결정하여 스케줄링 테이블에 등록한다.

앞서 언급한 초기화 과정에서는 SDT 전체에 대한 초기화 과정 이후에 전체 관리 객체들에 대해서 초기화가 수행되는데, 동작 중 변경의 경우에는, 전체가 아닌 해당 MO들에 대해서만 변경이 이루어진다.

2. 에이전트 요청에 따른 속성 값 반환

그림 10은 에이전트로부터 요청을 받고 이를 검색하여 반환하는 예를 나타낸다. RRS는 현재 MOT에 저장된 값을 바로 돌려준다.

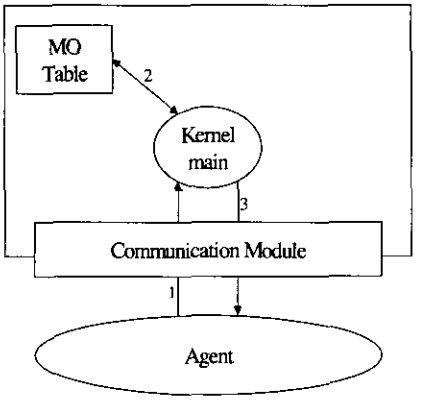


그림 10. 에이전트로부터 요청된 속성의 처리 과정

① 에이전트가 특정 관리 객체에 대한 속성 값을 요청한다.

② 커널은 MOT를 검색하여 속성 값을 얻는다.

③ 결과 값을 에이전트에게 반환한다.

3. 속성값 생성 및 저장

시뮬레이션 커널이 동작하게 되면, SDT에 정의된 값들에 의해 속성 값들을 생성하고 이것을 MOT에 저장해야 한다. 다음 그림 11은 이 과정을 보이고 있다.

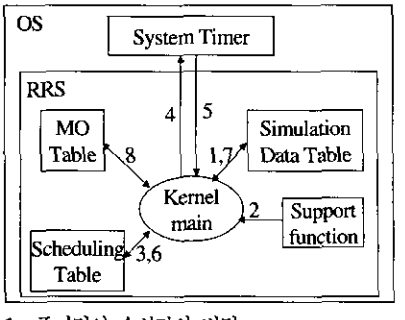


그림 11. 주기적인 속성값의 변경

① SDT가 초기화되었거나 속성값을 변경한 발생한 경우, SDT로부터 속성값을 생성시키기 위한 정보들을 가져온다.
② ①의 정보에 따라 해당 지원 함수를 실행시켜 다음 발생 시간을 구한다.
③ 구해진 발생 시간을 스케줄링 테이블에 등록한다. 이러한 스케줄링 테이블은 시간에 따라 순서적으로 정렬되어 있고, 상대적 타이머(relative timer)를 이용한다.
④ 맨 처음에 변경해야 하는 속성 값인 경우, 남은 시간을 OS의 타이머로 알린다.
⑤ 주어진 시간이 경과한 후 OS의 타이머로부터 시그널(signal)이 들어온다.
⑥ 스케줄링 테이블을 검색하여 해당 속성을 찾는다.
⑦ SDT로부터 랜덤 값을 발생시킬 지원함수를 찾는다.
⑧ 지원함수를 이용, 속성값을 생성하여 MO테이블의 속성 값을 변경한 후 ②로 돌아간다.

4. 통고의 생성 및 전달

속성의 경우 발생 시간이 되면 지원 함수를 실행하여 그 결과 값을 MO 테이블에 저장하는 것에 반하여, 통고는 발생 시간이 되면 이를 능동적으로 에이전트에게 전달해야 한다. 이에 따라 통고를 처리하는 과정은 속성의 처리 과정과 다르게 되는데, 그림 12에서 이를 보이고 있다.

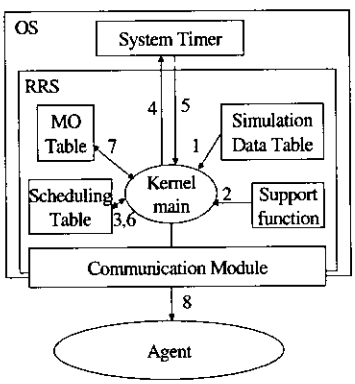


그림 12. 통고의 처리

① SDT가 초기화되었거나 통고가 발생한 경우, SDT로부터 통고를 발생시키기 위한 정보들을 가져온다.
② ①의 정보에 따라 해당 지원 함수를 실행시켜 다음 발생 시간을 구한다.
③ 구해진 발생 시간을 스케줄링 테이블에 등록한다. 이러한 스케줄링 테이블은 시간에 따라 순서적으로 정렬되어 있고, 상대적 타이머(relative timer)를 이용한다.
④ 맨 처음에 실행해야 하는 통고인 경우, 남은 시간을 OS의 타이머로 알린다.
⑤ 주어진 시간이 경과한 후 OS의 타이머로부터 시그널(signal)이 들어온다.
⑥ 스케줄링 테이블을 검색하여 해당 통고를 찾는다.
⑦ MO테이블로부터 통고와 관련된 정보를 찾는다.
⑧ 이를 에이전트에게 전송한 후 ②로 돌아간다.

Ⅳ. 확장된 GDMO 컴파일러

사용자는 SDT 정보들을 생성하기 위해서 기존의 관리 객체 정의와 별개로 각 관리 객체의 동작을 기술할 수 있어야 한다. 이러한 SDT 정보를 기술하고 처리하는 방안은 여러 가지가 있는데, 그 중 하나는 기존의 GDMO 문법[8]을 확장하여 이를 기술하고, 또한 기존 GDMO 컴파일러를 확장시켜 처리하는 방법이다.

1. GDMO 문법의 확장

사용자는 표준 GDMO 문법에 시뮬레이션 신택스(syntax)을 포함한, 확장된 GDMO 문법을 이용하여 시뮬레이션하고자 하는 실제 자원의 행동을 표현할 수 있다. 원래 GDMO 문법은 관리되는 객체를 9개의 템플릿(template)으로 나누어 정의하도록 하였다. 이중에서 관리 객체 템플릿은 관리 객체의 구성 요소를 정의하며, 이때 정의되는 구성 요소는 부모 클래스(class)와 관리 객체에 포함되는 패키지(package)들이다. 패키지 템플릿은 관리 객체 템플릿에서 참조하는 패키지의 구성 요소들을 정의하고, 여기에는 속성과 속성 그룹(attribute group), 동작(action), 통고 등이 있다. 이외에 속성 템플릿, 속성 그룹 템플릿, 동작 템플릿, 통고 템플릿, 파라미터(parameter) 템플릿, 네임바인딩(namebinding) 템플릿, 행동(behaviour) 템플릿이 있다.

앞서 언급한 테스트베드에서는 수행할 시뮬레이션 대상을 관리 객체의 속성과 통고로 제한하였다. 따라서, 확장된 GDMO 문법에서는 관리 객체에 포함되는 속성과 통고를 시뮬레이션할 수 있도록, 관리 객체의 속성과 통고를 기술하는 패키지 템플릿에 시뮬레이션을 위한 문법을 추가하였다. 그림 13

은 확장된 GDMO의 패키지 템플릿에 대한 문법을 나타낸다.

```
<package-label> PACKAGE
      [BEHAVIOUR<behaviour-definition-label>
                        [, <behaviour-definition-label>]* ; ]
      [ATTRIBUTES       <attribute-label> propertylist [<parameter-label>]*
                        [, <attribute-label> propertylist [<parameter-label>]* ]* ;
]
      [ATTRIBUTE GROUPS     <group-label> [<attribute-label>]*
                        [<group-label> [<attribute-label>]*]* ; ]
      [ACTIONS    <action-label> [<parameter-label>]* [, <action-label>
                  [<parameter-label>]*]* ; ]
      [NOTIFICATIONS
                  <notification-label> [period-definition <parameter-label>]*
                  [, <notification-label> [period-definition  <parameter-label>]*]*
; ]
[REGISTERED AS object-identifier] ;

supporting productions
propertylist   →    [REPLACE-WITY-DEFAULT]
            [DEFAULT VALUE          value-specifier]
            [INITIAL VALUE          value-specifier]
            [PERMITTED VALUES       type-reference]
            [REQUIRED VALUES        type-reference]
            [RANDOM VALUES              [random-value-specifier] ]
            [get-replace]
            [add-remove]
period-definition   →    RANDOM VALUES [random-function-definition]
value-specifier     →    value-reference !
                         DERIVATION RULE <behaviour-definition-label>
random-value-specifier →    period-function-specifier
                  generate-function-specifier
get-replace    →    GET | REPLACE | GET-REPLACE
add-remove     →    ADD | REMOVE | ADD-REMOVE
period-function-specifier          → PERIOD random-function-definition
generate-function-specifier → GENERATE random-function-definition
random-function-definition  → EXPONENTIAL <real-value>
                  | CONTINUOUS UNIFORM <real-value> <real-value>
                  | DISCRETE UNIFORM <integer-value> <integer-value>
                  | BINOMIAL <integer-value> <real-value>
                  | POISSON <real-value>
                  | GAUSSIAN <real-value> <real-value>
```

그림 13. 확장된 GDMO의 패키지 템플릿 문법

이 중 패키지 템플릿에서 속성과 통고를 실제 값이 아닌, 시뮬레이션 값을 사용한다는 것을 명시할 수 있다. 따라서, 사용자는 패키지 템플릿을 기술하면서 속성과 통고를 시뮬레이션 한다는 것과, 시뮬레이션에 사용하는 주기 및 생성 함수의 종류와 각각에 대한 파라미터를 정의할 수 있다. 또한 주기 및 생성 함수를 기술하지 않은 경우는 테스트베드의 운용 중에 GUI를 통해 사용자가 직접 지정할 수 있다.

그림 14는 확장된 GDMO로 기술된 패키지 템플릿의 예를 보여준다. 여기서는 CurrentUser라는 속성을 시뮬레이션 값으로 사용한다는 것을 선언하였고, 주기 함수로 Exponential 함수를, 값 생성 함수로 Discrete Uniform 함수를 사용한다고 선언하였다. 또한 attributeValueChange라는 통고를, 시뮬레이션으로 생성시켜 이용한다고 선언하였으며, 주기 함수는 시뮬레이터에서 초기화하도록 하기 위해 선언하지 않았다.

```
samplePackage              PACKAGE
    ATTRIBUTES
        ObjectId           GET,
        CurrentUser        GET-REPLACE
                RANDOM VALUES PERIOD EXPONENTIAL 20
                GENERATE DISCRETE UNIFORM 10 100,
        CurrentProcess     REPLACE-WITH-DEFAULT  GET-REPLACE;
    NOTIFICATIONS
        objectCreation,
        objectDeletion,
        attributeValueChange RAMDOM VALUES;
REGISTERED AS   { samplePackage 1 };
```

그림 14. 확장된 GDMO 문법으로 기술된 패키지 템플릿 예

2. GDMO 컴파일러의 확장

기존의 GDMO 컴파일러는 사용자가 기술한 GDMO를 입력으로 하여 에이전트에서 이용할 관리 객체 코드를 생성한다. 그러나 RRS에서는 실제 자원을 대신하여 시뮬레이션 하는 동작에 관련된 정보 역시 생성되어야 한다. 따라서 확장된 GDMO 컴파일러는 다음 그림 15와 같이 사용자의 입력에서 추가된 문법을 분리하고, 이를 이용하여 SDT 정보를 생성한다.

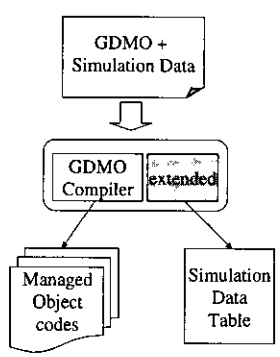


그림 15. 확장된 GDMO 컴파일러의 동작

실제로 시뮬레이션에 관련된 부분은 SDT 정보뿐이지만, 원래의 GDMO 컴파일러의 출력인 관리 객체 코드도 같이 이용할 수도 있으며, 이런 경우에는 망 관리시스템의 에이전트 생성도 동시에 수행할 수 있다. 확장된 GDMO 컴파일러는 사용자가 기술한 문서를 받아들여 원래의 GDMO 부분과 시뮬레이션 관련 부분을 분리하여 내부 클래스에 저장한다. 이와 같이 분리된 데이터를 바탕으로 관리객체 코드와 SDT 파일을 생성하게 된다.

그림 16은 확장된 GDMO 컴파일러의 전체적인 구조를 보이고 있다. 여기에는 내부적으로 전체적인 동작을 수행하는 Main 루틴과 초기화 루틴, GDMO

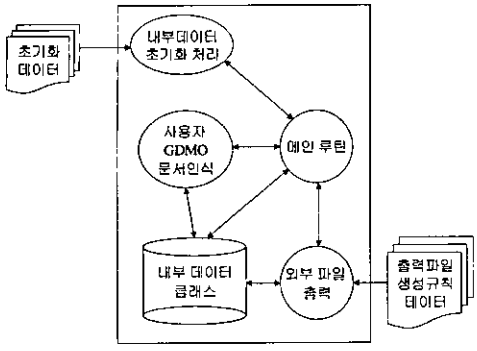


그림 16. 확장된 GDMO 컴파일러의 구조

문서 인식 루틴, 내부 데이터 클래스, 외부 파일 출력 루틴이 있다.

Main 루틴에서는 확장된 GDMO 컴파일러의 전체 동작을 관장하며, 각각의 내부 요소들과 상호 동작한다. Main 루틴의 주요한 동작은 ① 내부 데이터의 초기화와 ② 각각의 내부 요소의 동작 순서 유지, ③ 각 내부 요소에서 발생한 에러 처리 등이다. 내부 데이터 초기화 루틴에서는 확장된 GDMO 컴파일러가 수행하기 위해 필요한 초기화 파일들을 읽고, 그 정보를 Main 루틴에 전달하는 역할을 수행하고, GDMO 문서 인식 루틴에서는 ① 사용자가 기술한 문서를 읽어서 이를 해석하는 역할과 ② 해석된 데이터를 내부 데이터 클래스에 저장하는 역할, ③ 에러 발생 시 Main 루틴에 보고하는 역할, 그리고 ④ 사용자 문서에 대한 처리가 끝났을 때, 처리 결과를 Main 루틴에 보고하는 역할을 수행한다.

내부 데이터 클래스에는 사용자의 GDMO 문서에서 인식한 데이터들을 저장하고, 외부 파일 출력 루틴에서는 사용자 문서에서 인식된 결과를 이용하여 필요한 관리 객체 클래스 코드와 시뮬레이션 데이터를 생성하는 역할을 수행한다. 또한 초기화 데이터에서는 컴파일러를 동작시키기 위해 필요한 초기화 정보를 유지하는데, 여기에는 기존에 정의된 관리 객체, 속성, 신택스의 정보등이 포함된다. 마지막으로 출력 파일 생성 규칙 데이터에는 내부 데이터 클래스에 저장된 정보를 이용하여, 관리 객체 클래스와 시뮬레이션 데이터를 생성하는 규칙을 저장한다. 확장된 GDMO 컴파일러의 수행 과정은 그림 17과 같다.

① 확장된 GDMO 컴파일러는 초기화 데이터 파일을 읽고, 초기화 데이터 파일의 정보를 이용하여 내부 데이터를 초기화 한다.

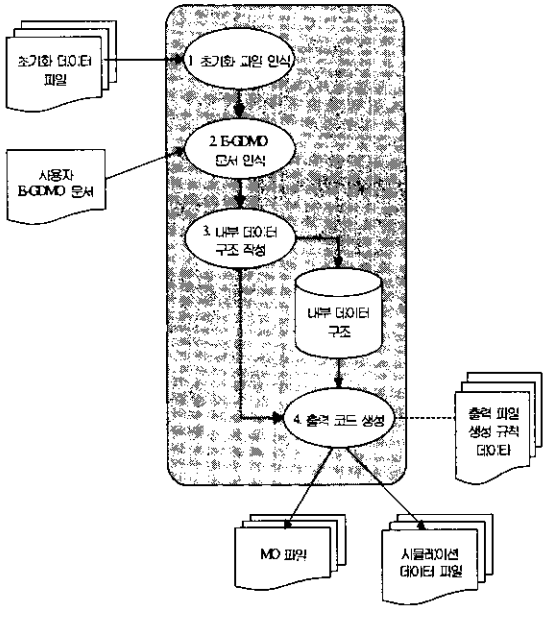


그림 17. 확장된 GDMO 컴파일러의 수행과정

② 컴파일러는 사용자가 작성한 GDMO 문서를 읽고, 사용자 문서에 포함된 정보들을 인식한다.

③ 사용자의 문서에서 인식된 정보를 이용하여 내부 데이터 구조에 해당 정보들을 저장한다.

④ 확장된 GDMO 컴파일러는 내부 데이터 구조에 저장된 정보를 이용하여, 출력 파일 생성 규칙에 따라 관리 객체 클래스 파일과 SDT를 생성한다.

V. RRS를 이용한 망 관리 시스템 평가 방안

1. 실제 자원의 동작 특성 수집

무엇보다도 RRS는 실제 자원의 동작을 시뮬레이션 해야 하므로 이에 관련된 실험 데이터를 생산하는게 중요하다. 이것은 오랜 시간에 걸친 실제 자원에 대한 관찰과 기록으로 얻을 수 있는데, 실제 자원이 개발되기 전에는 비슷한 기능을 갖는 다른 자원에 대한 기록으로도 가능하다. 또한 실제 자원이 개발된 후에도 이러한 동작 특성은 기록으로 계속 유지하는 것이 망에 대한 기능 및 성능 개선이나 관리 시스템에 대한 개선에 큰 도움이 될 수 있다.

2. 에이전트 성능 평가

RRS를 이용하면 실제 자원 구현 전에도 관리 에이전트의 기능과 성능 평가가 가능하다. 이때의 주 평가 대상은 실제 자원에서 사건들이 발생할 때 이를 받아 관리자에게 통고하는 에이전트의 처리 능력이 된다. 따라서 임의의 사건을 발생시키고, 이에

대한 처리 능력을 관찰할 수 있다. 이때에는 에이전트가 하나의 RRS를 관리할 수도 있고, 여러 RRS를 관리할 수도 있다. 그림 18은 RRS를 이용한 에이전트 성능 평가 구성을 보여준다.

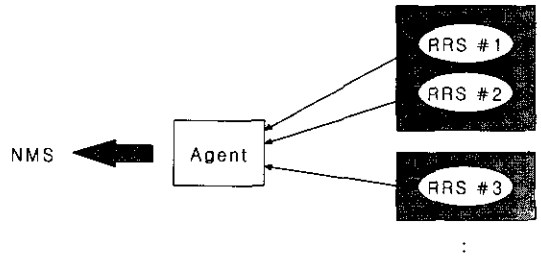


그림 18. RRS를 이용한 에이전트 성능평가

3. 관리 시스템의 평가

망 관리시스템에 대한 평가는 그 망에 속한 전체 요소들에 대한 평가 작업이 된다. 이때에는 에이전트에 대한 능력과 망 관리 시스템에 대한 능력을 같이 관찰할 수 있게 되는데, 이러한 평가를 바탕으로 관리시스템의 기능 평가나 개선 작업이 수행될 수 있다. 이때 주로 보게될 항목은 관리시스템에서 요구한 속성 값의 반환이나 동작에 대한 에이전트의 처리 능력, 각 에이전트들이 전달한 통고에 대한 관리 시스템의 처리 능력 등이다. 특히 본 논문에서는 확장된 GDMO 컴파일러를 이용함으로써 RRS 기능과 에이전트 기능을 같이 갖는 시스템을 생성시킬 수 있다. 그림 19는 RRS 기능을 포함한 에이전트를 이용하여 NMS를 평가하는 구성을 보여준다. 한 시스템 내에는 여러 에이전트가 상주할 수 있고, 각 에이전트에는 여러 RRS가 포함될 수 있다. 이러한 구성은 실제 망 요소들 없이도 가상 망을 구성하고 NMS의 운용 및 성능 평가를 가능하게 한다.

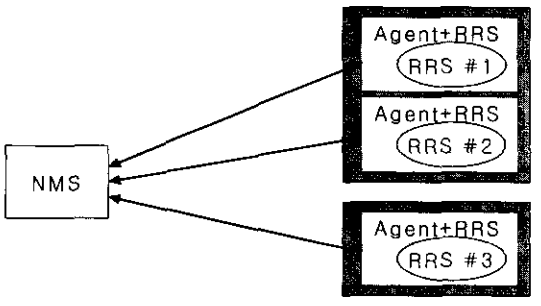


그림 19. RRS 기능을 포함한 에이전트를 이용한 NMS 성능 평가를 이용한 에이전트 성능평가

Ⅵ. 결론

본 논문에서는 TMN 상에서 실제 자원이 없이도 사용자가 정의한 실제 자원의 동작 특성에 따라 그 행동을 대신할 수 있는 RRS를 제안하였다. 또한 이러한 RRS를 지원하기 위해 기존의 GDMO를 확장하고, 이를 위한 확장된 GDMO 컴파일러를 제안하였다. 제안된 RRS는 관리 객체의 속성과 통고를 고려하여 설계되었으며, 사용자가 설정한 시뮬레이션 관련 정보에 따라 속성 값을 변경시키고, 통고를 발생시키도록 되어 있다. 이러한 RRS가 구현되면, 실제 자원 개발 전에도 망 관리 에이전트와 망 관리 시스템의 개발이 원활히 이루어 질 것이며, 또한 가상적인 망 상황을 구성하여 망 관리 시스템의 기능과 성능 테스트를 수행할 수도 있을 것이다. 현재, 각 요소 별로 세부 설계가 진행중이며, UNIX의 사용자 영역에서 구현할 예정이다.

참 고 문 헌

[1] ITU-T M.3010, "Principles for a Telecommunication Management Network"

[2] ISO7498-4/ITU-T X.700, "OSI Basic Reference Model Part 4: Management Framework"

[3] ISO10040/ITU-T X.701, "Systems Management Overview"

[4] ISO9595/ITU-T X.710, "Common Management Information Service Definition"

[5] ISO9596-1/ITU-T X.711, "Common Management Information Protocol Specification"

[6] ISO10165-1/ITU-T X.720, "Management Information Model"

[7] ISO10165-2/ITU-T X.721, "Definition of Management Information"

[8] ISO10165-4/ITU-T X.722, "Guidelines for the Definition of Managed Objects"

[9] ISO10165-5/ITU-T X.723, "Generic Management Information"

[10] ISO10164-5/ITU-T X.734, "Event Management Function"

[11] George Pavlou, "The OSIMIS Platform: Making OSI Management Simple", Integrated Network Management IV, 1995

[12] George Pavlou, "High-Level Access APIs in

the OSISMIS TMN Platform: Harnessing and Hiding", 2nd conference on Intelligent Services and Network Management, 1994

[13] George Pavlou, "Experience of Implementing OSI Management Facilities", Integrated Network Management II, 1991

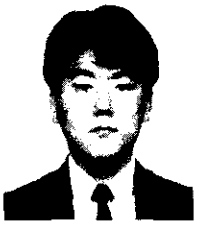
[14] G. Pavlou, J. Cowan, J. Crowcroft, "A Generic Management Information Base Browser", *IETF* TC6/WG6.5, 1992

송 병 권(Byung-kwen Song) 정회원

1984년 : 고려대학교 전자공학과(공학사)
1986년 : 고려대학교 대학원 전자공학과(공학석사)
1995년 : 고려대학교 대학원 전자공학과(공학박사)

1984년~1997년 : 삼성종합기술원 선임연구원
1995년~현재 : 서경대학교 정보통신공학과 교수
<주관심 분야> High-speed Network, 분산처리시스템, Mobile computing

김 건 웅(Geonung Kim) 정회원



1990년 : 고려대학교 전자전산공학과(공학사)
1994년 : 고려대학교 대학원 전자공학과(공학석사)
1998년 : 고려대학교 전자공학과(공학박사)

1999년 9월~현재 : 목포해양대학교 해양전자·통신공학부 전임강사
<주관심 분야> 통신망 프로토콜, 망관리 시스템, 지능망, 정보망, 멀티미디어 통신

진 명 숙(Myung-sook Jin) 정회원

1990년 : 고려대학교 전자전산공학과(공학사)
1992년 : 고려대학교 대학원 전자공학과(공학석사)
1997년 : 고려대학교 전자공학과(공학박사)

1996년~2001년 : 경인여자대학 멀티미디어 정보전산학부 조교수
2001년~현재 : 명지전문대학 정보통신과 조교수
<주관심 분야> 분산시스템, 멀티미디어 컴퓨팅, 객체지향시스템