韓國 OA學會 論文誌
第4卷 第4號, 1999.12.

99-4-4-3-10

TMN기반의 SMF설계를 위한 효율적인 자원 감시 기법

정 현 식*, 전 상 훈**

Efficient Real Resource Monitoring Methods for Design System Management Function based on TMN

Hyun-Sik chung*, Sang-Hoon Jeon**

요 약

TMN체계에서 GDMO(Guidelines for the Definition of Managed Object) 에이전트 플랫폼(Agent Platform)은 망 요소의 운영 상태와 자원들을 GDMO 표준에 따라 관리객체(Managed Object : MO)로 모델링 하고, 자원들의 상태를 유지하면서 관리자(manager)로 부터의 망관리 기능 요구에 따라 조작된다. 이러한 GDMO 에이전트의 기본 기능을 지원하기 위해 13가지의 시스템 관리 기능(System Management Function : SMF)이 구현되어져야 한다. 이 13가지의 시스템 관리 기능들 중 사건-보고-관리 기능(Event-Report-Management Function)은 모든 시스템 관리 기능에 관련되어진다. 그 이유는 지속적인 장비들과의 통신을 통하여 관리자(manager)측으로 즉각적인 자원들의 상태에 대한 보고가 일어나야 하기 때문이다. 본 논문에서는 효율적인 사건 보고 관리 기능을 위하여 장비(real resource)들과의 통신을 위한 몇 가지 방법에 대한 분석을 행하고 여기서 제시되는 문제점 향상을 위해 RRMC(Real Resource Message Coordinator)구조의 새로운 자원 감시 기법을 제안한다.

Abstract

GDMO Agent platform based on TMN modeling operating status of network elements and resources to Managed Object as GDMO standard and controled by telecommunication management function requests of manager supporting status of resources. To supporting basic function of these GDMO agent, 13 kinds of system management function has to be implemented. Event-Report-Management function related to all system management functions. The reason that rapidly reporting has been performed as status of resources to manager by continuously communication with real resources. In this paper, a number of approach has been analyzed to communication with real resources for effective event-report management function and we proposed new real resource monitoring schemes of RRMC(Real Resource Message Coordinator) architecture for improving these resulted problem.

*경도대학 사무자동화과 전임강사
*경동정보대학 전자정보과 전임강사
논문접수:1999.11.15. 심사완료:1999.12.10.

Ⅰ. 시스템 관리 기능의 개요

TMN관리 기능들은 목적에 따라 다섯 가지의 관리 기능 영역(Managed Functional Areas : MFAs)으로 나누어지며 ITU-T X.700에서 권고하고 있다. 또한 TMN에이전트는 OSI에서 정의하는 5가지 망 관리 기능 분야(FCAPS)의 기능구현을 위해 13개의 시스템 관리 기능을 제공한다[1]. 5가지 망 관리 기능분야와 13개의 시스템 관리 기능은 아래와 같다.

■ 망 관리 기능 분야(FCAPS)

- 장애 관리(Fault Management)
- 구성 관리(Configuration Management)
- 과금 관리(Accounting Management)
- 성능 관리(Performance Management)
- 보안 관리(Security Management)

시스템 관리 기능은 위의 다섯 가지 영역들 내 기능의 중복을 피하기 위해서 13가지의 관리 기능으로 세분화하여 아래와 같이 정의한다.

■ 시스템 관리 기능(System Management Function)

- 객체 관리 기능(Object-Management Function)
- 상태 관리 기능(State-Management Function)
- 관계 관리 기능(Relationship-Management Function)
- 경보보고 기능(Alarm-Reporting Function)
- 사건보고 관리 기능(Event-Report Management Function)
- 로그 제어 기능(Log-Control Function)
- 보안경보보고 기능(Security-Alarm-Reporting Function)
- 보안감시 추적 기능(Security-Audit-Trail Function)
- 접근제어 관리 기능(Access-Control Management Function)
- 과금 측정 기능(Accounting-Meter Function)
- 부하 감시 기능(Workload Monitoring Function)
- 시험 관리 기능(Test Management Function)
- 요약 기능(Summarization Function)

각각의 시스템 관리 기능 표준들은 시스템 관리 기능에 대해 기능적으로 정의하고 있으며, 시스템 관리 기능과 CMIS(Common management information service)에 의해 제공되어지는 서비스들 사이의 연결(mapping)을 제공한다[2]. 이들의 관계를 (그림 1)에서 보이고 있다. 그림에서 보듯이 각 시스템 관리 기능은 한 개 이상의 시스템 관리 기능 영역을 위한 정보의 관리를 위해서 어떤 접근의 부류를 제공한다. 시스템 관리 기능들은 서로 복잡하게 관련이 되어있으며 (그림 1)은 직접 CMISE 서비스의 일부를 사용하는 시스템 관리 기능들을 보여준다.

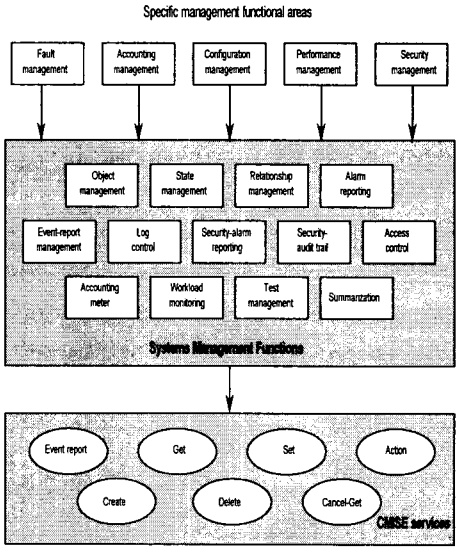

그림 1. 시스템 관리 기능 전체 개념도

Ⅱ. 사건보고 관리 기능

사건보고 관리 기능은 X.734/ISO 10164-5에서 제공한다. 관리 객체들의 정의에 독립적으로 관리객체로부터 사건 보고의 전송 제어를 관리자에게 허용해준다. 관리 객체들로부터의 보고가 제어되어지는 수단으로써 사건-추

진 판별자(Event-Forwarding Discriminators)라고 불리는 객체가 추가된다. 이 판별자(discriminator)는 다른 객체들에 관련된 관리 동작을 제어하기 위해 시스템에 허용되는 관리객체이다. 사건-전 판별자는 잠재적인 사건 보고상의 동작중 하나이다. 부수적으로 필터(filter)가 정의되어질 수 있는데 그것은 보고되어질 사건들을 명시한다. 사건보고 관리 기능은 사용자에게 사건들의 선택을 제어할 수 있게 하며, 관리자의 특별한 목적지에 그들의 분배를 허락한다. 즉, 사용자에게 사건-추진 판별자 관리 객체를 만들 수 있게 하며, 이것은 사건-전 판별이 발생하였을 동안 시간 기간뿐만 아니라 사건 보고들까지도 원격적으로 전달이 되어질 관리객체 통지의 기준을 관리자가 만들고 선택할 수 있도록 정의해준다[3, 4].

2.1 사건보고 처리 절차

(그림 2)에서 보듯이 전체적인 사건보고 처리 절차는 아래와 같다.

① 적어도 관리 시스템 내의 한 개의 관리객체는 사건보고로 추진되어지는 통지를 발생할 수 있다. 통지는 사건들이 발생했을 경우 관리객체들에 의해 발생된다.

② 사건 감지-처리 모듈(Event detection and processing)은 지역에서 발생하는 통지들을 받아서 잠재적인 사건 보고들은 형성한다. 잠재적 사건 보고들은 사건 시간, 객체 클래스, 인스턴스와 같은 통지 내의 모든 정보들을 담고 있다.

③ 잠재적인 사건 보고들은 모든 지역 시스템내의 사건-추진 판별자들에게 분배되어진다. 판별자 역시 하나의 관리객체이며 잠재적인 사건보고 객체의 행동을 유발시킨다.

④ 판별자는 실제적인 사건 보고로서 특별한 목적지까지 사건 보고가 추진되어야 하는지를 결정한다. 판별자는 판별자 구조(Discriminator Construct)를 담고있는데 그것은 잠재적인 사건 보고가 추진되어지는 것이 적절하기 위해서 만족해야 하는 특성을 명시하고 있다. 즉 이것은 부수적으로 필터링 기능이다. 또한 추진을 위해 선택되어지는 사건 보고들의 간격을 결정하는 스케쥴링 기능도 가지고 있다.

⑤ 사건보고 판별자는 다른 관리객체와 같이 상태들과 속성들을 지니는 관리객체이다. 사건보고 관리 기능은 사건-추진 판별자의 속성을 읽고 수정할 수 있다. 그림으로써 사건보고 판별은 초기화, 종료, 지연, 재 개시되어질 수 있다. 특별한 스케쥴링 과 필터링 정보는 읽혀지고 수정되어질 수 있다. 그후 추진이 판별된 사건보고는 지역시스템이나 요구된 시스템이 직접적으로 전해진다.

⑥ 또한 다른 관리객체와 같이, 사건보고-판별자 객체들은 통지를 방출한다. 이런 통지들은 모든 사건-추진 판별자에 의해서 잠재적인 사건보고로 처리되어진다.

Ⅲ. 자원과의 통신

3.1 에이전트 코어(Agent Core)

에이전트 코어는 그림 3과 같은 기능을 수행하는 GDMO 에이전트 플랫폼의 핵심이 되는 부분으로 실제 OSI 관리 모델에서 제시하는 5가지 시스템 관리기능 영역(FCAPS)에 대한 13개의 시스템 관리 기능(SMF)을 수행하기 위한 모듈과 관리자와의 통신에 필요한 통신 프로토콜과의 인터페이스를 위한 모듈, 그리고 MIB/MIT와의 인터페이스를 위한 모듈로 구성된다[5, 6]. 이러한 에이전트 코어 부분의 모듈들은 여러 개의 클래스로 설계·구현되며, 각각의 애트리뷰트, 액션(action), 통지(notification)등과 관련된 정보를 교환한다. 에이전트 코어는 이러한 각각의 기능 모듈들을 이용하여 에이전트 시스템의 최초 설정시 TMN 관리 기능을 초기화시킨 후 관리자로 초기화 보고를 하게 된다.

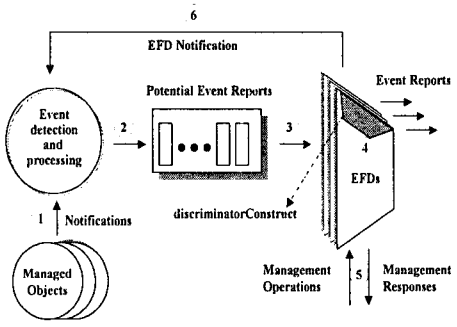

그림 2. 사건 보고 기능 처리 절차

에이전트 코어에서 망관리 통신 프로토콜과의 인터페이스는 어소시에이션 서비스(Association service), 매니저 핸들(Manager handle), 송신 제어 객체(Send control object) 모듈을 통해 이루어진다. 어소시에이션 서비스는 관리자와 통신을 위한 초기 어소시에이션 연결 확립 및 해제에 대한 기능을 가진다. 매니저 핸들은 관리자와 통신을 할 때 AE- Title(Application Entity Title)이나 PSAP(Presentation Service Access Point)주소를 제공한다. 송신 제어 객체는 관리기능 요청(CMIP Request) 결과에 대한 응답이나 통지(notification)를 위한 통신 파라메타들의 제어를 위해 사용된다.

Naming 서비스 모듈은 MIB/MIT 구성시 DN(Distinguished Name)과 RDN(Relationship Distinguished Name)에 관계되는 부분을 처리하기 위해 사용된다.

MO machine 과 MO factory는 초기 에이전트 시스템 설정시 관리객체의 생성과, MIB/MIT의 구성 및 관리를 위해 사용된다.

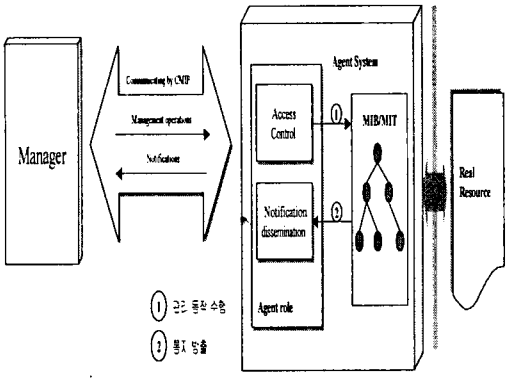

그림 3. GDMO Agent Platform의 개념도

3.2 자원 감시 기법(Real Resource Monitoring Method)

자원들의 정보는 커널들의 가상 메모리 혹은 장치 드라이버를 읽거나 IPC 메카니즘을 이용한 다른 사용자 영역 프로세스와의 통신 그리고 proxy 관리의 경우 통신 프로토콜을 사용한 원격 시스템으로의 통신에 의해서 접근되어 질 수 있다.

자원들로부터의 사건 발생시 이를 감시하기 위한 기법으로 해당 관리 객체와 연관된 자원들과의 통신을 위해 UCL(University Collage London)에서 개발된 OSI 망 관리 시스템인 OSIMIS[7,8,9,10]의 구조를 조사해 보았다. OSIMIS에서는 폴링(polling) 기법과 사건 구동(event driven) 기법 그리고 외부 요구(external request) 기법등 3가지 메커니즘을 가진다.

3.2.1 폴링 기법(Polling Method)

폴링 기법은 시간 구동 기법(time driven method)이라고도 불리며, 에이전트 코어에서 자원들의 임계값(threshold)과 연관된 관리 객체 내 각각의 애트리뷰트 값(attribute value)을 정해진 시간 간격을 가지고 주기적으로 검색하여 임의의 애트리뷰트 값이 해당 임계값을 초과하였을 경우 그때 관리자에게 사건 보고를 위해 관리 객체내의 사건 유발 메쏘드를 호출하게 한다. 이 기법은 결코 필요로 되지 않는 많은 데이터들을 주기적으로 검색하여야 하기 때문에 다소 비효율적이다. 하지만 자원 내에서 몇몇 특별한 사건이 발생할 경우 통지가 일어나야 한다면 유일한 방법일 것이다. 아래의 그림은 폴링을 위한 해당 관리객체의 등록 절차를 나타낸 것이다.

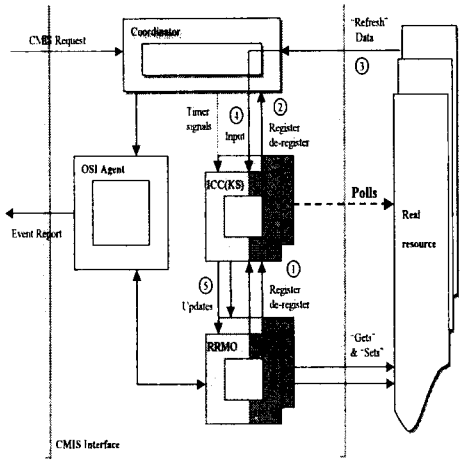

그림 4. 자원과의 통신을 위한 등록 절차

그림 4에서 ICC(Internal Communications Control)는 코디네이터와 함께 포트에 자원관리객체(RRMO : Real Resource Managed Objects)를 소켓으로 등록시키는 역할을 담당한다.

① RRMO가 CMIS M-CREATE 요청을 통해 생성되어졌을 때, 그것은 자신을 ICC 객체에 등록시킨다. 이 경우 만약 ISODE와 관련된 첫번째 자원관

리객체의 등록이라면 ICC는 활동중인 ISODE상의 프로세스들에 접속되어지기 위해서 UDP소켓을 초기화시킨다.

② 또한 자원관리객체의 활동 상황을 통지하기 위해서 코디네이터에도 "communication endpoint"로 등록시킨다.

③ 주기적인 SIGALRM에 의해 ISODE 프로세스로부터 UDP소켓에 메세지가 도착하였을 경우 코디네이트는 이를 감지한다.

④ 코디네이트는 메세지를 해당 ICC로 넘긴다.

⑤ ICC는 관련된 RRMO에게 들어온 정보를 넘긴다.

각각의 관리객체 정의시 자원과의 통신을 위한 메쏘드의 마지막 부분에 아래의 함수가 삽입되어진다.

schedule_poll(Interval, NULLCP, True);

이 함수는 유닉스의 SIGALRM을 이용하여 지정된 주기(interval)가 지나면 해당 신호 처리함수를 호출하도록 되어있다. 이 때 호출되는 처리함수는 해당 관리 객체내의 애트리뷰트 값에 대한 최신 정보를 가져오는 함수이다. 이 때 가져온 값이 임계값을 초과하였다면 사건보고가 유발될 것이다.

폴링에 의해 감시될 자원들이 여러 개일 경우 이들은 코디네이트에서 폴링 스케쥴에 의해 따로 관리가 되어진다.

3.2.2 사건 구동 기법(Event Driven Method)

사건 구동 기법은 자원들에서 임의의 사건이 발생하였을 경우, 전형적으로 IPC 메카니즘으로 유발되어 자원과의 통신이 초기화되어진다. 관리자에게로 보고가 끝나게 되면 연결이 자동으로 해제되어 비동기적으로 자원으로부터의 활동 상황을 받아들일 수 있다. 즉, 자원들에서 자발적으로 에이전트 코어로 사건보고를 하고, 에이전트 코어에서 관리자로 사건보고를 수행한다. 이 기법은 분산된 기계나 분산된 유닉스 프로세스 내에 있는 자원들의 구성에 적합하다. 통신을 위한 등록 절차는 폴링기법에서와 유사하며, 이는 ISODE 트랜스포트를 채용한 모든 사용자 프로세스들로부터 정보를 받기 위해 공통 포트를 경청한다. 그 이유는 자원에서 사건이 발생할 경우, 트랜스포트 프로토콜 코드는 UDP 메세지를 공통 포트에 방출하기 때문이다. 사실상 에이전트 시스템은 몇 개의 소켓을 경청하는 것이다. 이러한 모든 소켓들의 등록 및 해제는 코디네이트에 의해 집중 관리되어진다. 그렇지 않을 경우 하나의 소켓을 읽으므로 해서 전체 시스템이 봉쇄되어질 수 있다. 유닉스 "select" 시스템 호출을 이용하여 모든 관련된 소켓들을 등록시켜 경청할수 있다. 그리고 들어오는 메세지는 해당 관리 객체에 넘겨진다. 그림 5는 사건구동 기법을 위한 select 시스템 호출을 사용한 예이다.

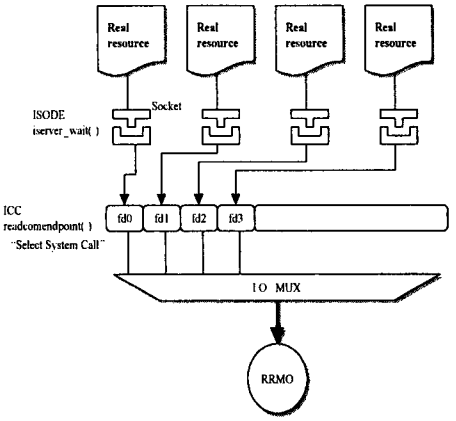

그림 5. "select" 시스템 호출을 이용한 사건구동 기법

그림 4의 ②에서 "communication endpoint"등록시 소켓으로부터 들어오는 메세지를 기다리기 위해 ISODE의 iserver_wait( )이 호출된다. 소켓에 메세지가 도착할 경우 소켓을 읽기 위해 ICC의 readCommEndpoint( )를 호출한다. 여기서 select 함수가 불려지며 그림 5에서처럼 select함수는 SVR4와 4.3+BSD에서 입출력 다중화(I/O Multiplexing)를 가능하게 한다. 이 결과 해당 관리객체에 자원의 메세지가 반영되어진다.

3.2.3 외부 요구 기법(External Request Method)

스코핑(scoping)과 필터링(filtering)을 거친후 CMIS 요청이 에이전트에 받아들여질 경우 자원과의 통신이 유발되어진다. 이 기법은 자원과의 통신을 최소화시키며 외부 요청이 받아들여지기 전까지 자원과의 통신은 없게된다. 이 기법은 임계값 유발과 같은 데이타 의존적인 사건들을 구현하는데는 적합하지 않다. 즉, 이 기법은 주문형(on-demand) 전략이며 기본 MO 클래스의 메쏘드인 MO::get( )을 통해 요청한다.

Ⅳ. 분산 및 병렬처리를 위한 자원감시기법

위의 3가지 방식은 단일-쓰레드 실행 패러다임하에서 모든 내부(internal), 외부(external) 사건들을 선입선처리(First Come First Service) 방식으로 해당 관리객체와의 연결을 시도한다. 하지만 첫째, 이러한 방식으로는 관리객체의 수가 증가함으로써 야기되는 복수개의 자원으로부터 경보(alarm)-보고 및 복수개의 애트리뷰트 값에 대한 폴링 결과를 동시에 받아들일 수가 없게된다. 둘째, 현존의 OSIMIS의 구조는 동기형(synchronous)으로 설계가 되어졌기 때문에 매니저측에서 에이전트측으로의 CMIP 메쏘드의 호출(External Request)시 반드시 애트리뷰트 값이나 에러등의 결과가 리턴되어야만 한다. 이것은 그 결과를 리턴하기 전까지 전체 시스템을 블럭킹 상태로 만든다. 만일 스코핑을 통해 동시에 여러 개의 객체들이 접근되어질 경우 그 순간 다른 요청들은 대기상태로 놓이게 되어 시스템의 성능저하를 가져온다. 여기서의 대안은 멀티쓰레드 구조의 도입이 될 것이다. 물론 비동기 입출력으로 설계를 하면되지만, 비동기 입출력의 한가지 제약은 프로세스당 하나의 신호만이 존재한다는 것이다. 비동기 입출력을 위해 하나이상의 지정번호를 활성화하면, 신호가 전달되었을 때 어떤 지정번호가 그 신호에 상응하는 것인지 구분할 수 없다[11]. 셋째, 자원감시 기법 중 폴링 기법은 대부분의 시스템에서 경보 보고 기능과 같은 관리를 수행하기 위해 필수적일 것이다. 그러나 몇 개의 애트리뷰트 값에 대한 주기적인 폴링은 코디네이터 측에서의 스케쥴러에 의해 가능하지만, 실제적으로 예를들어 ATM 교환기와 같은 자원에 대한 망 관리를 구현할 경우 수백~수천개 혹은 그이상의 애트리뷰트 값을 주기적으로 폴링하여야 한다면, 현존의 단일 프로세스 실행 패러다임으로는 불가능하다.

Ⅴ. RRMC구조의 그룹핑 폴링 기법(Grouping Polling Method of RRMC)

폴링을 위해 등록되어지는 관리객체내의 애트리뷰트값들에 대해 각각의 쓰레드를 생성시켜 주어진 주기마다 자원을 감시한다면 수없이 많은 쓰레드가 생성이 되어 그로 인한 시스템의 부하가 증가할 것이다. 실제로 쓰레드의 수가 수십개이상이 되면 그로 인한 부하가 큰 것으로 알려져 있다. 또한 폴링을 위해 사용되어지는 유닉스의 SIGALRM신호는 하나의 프로세스 내에 하나만이 존재하기 때문에 폴링을 위해 애트리뷰트마다 하나의 쓰레드로 구현한다는 것은 역시 불가능한 이유가 된다.

본 논문에서는 이러한 문제점을 해결하기 위한 방안으로 RRMC구조의 새로운 자원 감시 기법을 제안한다. 이 구조에서 등록된 수많은 애트리뷰트들은 SIGALRM신호를 사용하는 대신에 감시를 위한 기본 주기로서 매 1초마다 주기적으로 각 애트리뷰트들에게 감시신호를 전달한다. 여기서 각 애트리뷰트들은 자신의 주기 즉, 5초라면 5번의 감시신호가 전달될 때 마다 해당 쓰레드를 생성시켜 실질적인 감시를 위한 메쏘드를 호출하고 삭제하는 과정을 반복하게된다. 이러한 스케쥴 정책은 매 감시신호가 들어올 때 카운터를 증가시켜서 이 값을 자신의 감시 주기로 나누고 그 몫이 0으로 나누어 떨어지면 자신의 감시를 위한 메쏘드를 호출한다. 또한, 같은 감시 주기를 가지는 애트리뷰트들과 각 애트리뷰트들의 감시 주기가 배수 관계에 있는 것들은 동시에 자원을 감시해야하므로 그룹핑하여 보낼 수 있다. 이러한 스케쥴 정책은 감시 스케쥴러에 의해 행해지며 감시자는 매 주기마다 동시에 감시되어질 자원들을 그룹핑하여 각각의 쓰레드를 생성시켜 감시 메쏘드를 호출하고 그 다음 삭제하므로서 분산 및 병렬처리를 가능하게한다. 그림 6은 RRMC구조의 그룹핑 폴링 기법을 나타낸다.

매 주기마다 형성되는 자원 감시를 위한 그룹들은 동적으로 쓰레드를 생성시키고 원하는 애트리뷰트의 최신값

을 읽어온후 삭제하므로 전체 시스템의 성능에 영향을 미치지 않고 다수의 자원을 감시할 수가 있다.

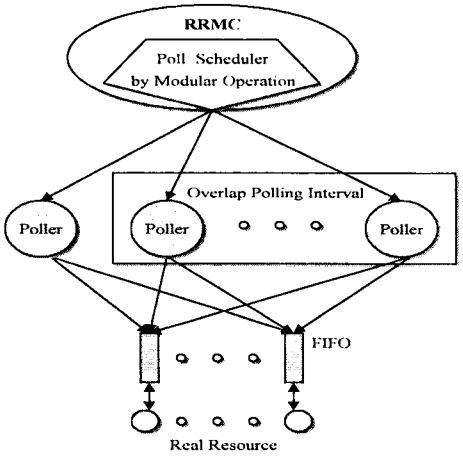

그림 6. RRMC구조의 그룹핑 폴링 기법

Ⅵ. 결 론

지금까지 기존의 OSIMIS 통신망 관리 시스템에서의 자원과의 통신 기법에 관하여 분석, 조사하였으며 여기에서의 병렬·분산처리시 문제가 되는 자원감시기법을 해결하기위한 방안을 검토해 보았다. 관리객체의 감시될 애트리뷰트값들을 어떤 식으로 감시해야 할 것인가를 신중히 고려하여 그 애트리뷰트값의 특성에 맞도록 즉, 에이전트 시스템의 부하를 최대한 줄이면서 효율적인 자원의 감시가 가능하도록 관리객체를 설계하는것이 중요하다. 본 연구에서는 대부분의 시스템에서 경보 보고 기능과 같은 관리를 수행하기 위해 필수적인 폴링기법에서 실제 교환기와 같은 망관리 시스템의 적용시 문제가 되는 스케쥴 관리 및 시스템의 성능저하를 해결하기 위한 방안으로 RRMC구조의 그룹핑 폴링 기법을 제안하였다. 이 기법을 이용함으로써 현실적인 TMN체계의 망관리시 자원과의 통신을 더욱 효율적으로 수행할 수 있다.

참고 문헌

[1] ITU-T Rec. X.730-745, "System Management Function", Jan. 1993.

[2] ITU-T Rec. X.711, "Common Management Infor- mation Protocol Specification for CCITT Application", 1991.

[3] 연차보고서, "ATM/B-ISDN 통신망 관리를 위한 TMN 체계의 GDMO Agent Platform 개발", (주)토미스/영남대학교 정보통신 연구소, 정보통신부, Mar. 1997.

[4] ITU-T Rec. X.720, "Information Technology-Open System Interconnection (OSI) Structure of Management Information Model", Jan. 1992.

[5] ITU-T Rec. X.722, "Guidelines for the Definition of Managed Object(GDMO)", Jan. 1992.

[6] Sloman, "Network and Distributed Systems Man- agement", ADDISON WESLEY, 1994.

[7] George Pavlou, Graham Knight, Kevin McCarthy, Saliim Bhatti, "The OSIMIS Platform: Making OSI Management Simple", 1994

[8] George Pavlou, Graham Kinght and Simon Walton, "Experience of Implementing OSI Management Facilities"., 1994

[9] George Pavlou, Thurain Tin, "A CMIS-capable Scripting Language and Associated Lightweight Protocol for TMN Applications"., 1994

[10] George Pavlou, Thurain Tin, "High-Level Acc- ess APIs in the OSIMIS TMN Platform: Harnessing and Hiding"., 1994

[11] W.Richard Stevens, "Advanced

Programing in the UNIX Environment", ADDISON WESLEY, 1992

저 자 소 개

정 현 식

1987년 2월 경일대학교 전산공학과 졸업(공학사)

1990년 2월 영남대학교 대학원 전자공학과 졸업(공학석사)

1997년 2월 영남대학교 대학원 전산공학과 박사수료

1999년 3월 ~현재 경도대학 사무자동화과 전임강사

관심분야: 컴퓨터 구조, 분산처리 시스템, 이동통신 데이터베이스

전 상 훈

1992년 2월 영남대학교 전산공학과 졸업(공학사)

1994년 2월 영남대학교 대학원 전산공학과 졸업(공학석사)

1998년 8월 영남대학교 대학원 전산공학과 박사수료

1999년 3월~현재 경동정보대학 전자정보과 전임강사

관심분야: 멀티미디어 시스템, 정보통신, 컴퓨터 구조