論 文

위임 모델에서의 네트워크 관리 스크립트 작성에 관한 연구

正會員 韓 順 姬* 正會員 李 基 鉉** 正會員 趙 國 鉉*

Network Management Script Construction in Delegation Model

Soon Hee Han*, Kee Hyun Lee**, Kuk Hyun Cho* *Regular Members*

要 約

네트워크 관리를 위한 원격 위임 모델은 관리 기능들을 관리자와 피관리자 간에 적절하게 분배함으로써 보다 신뢰성 있고 효율적인 네트워크 관리를 지원하고, 복잡하고 고속인 네트워크 내에서 네트워크 관리의 신뢰도를 높이기 위한 모델이다. 이 모델에서는 관리자는 관리 기술 언어로 미리 기술되어 있는 관리 프로그램의 실행을 피관리자에게 위임하며, 피관리자는 프로그램 내의 퍼리미티브들을 이용하여 관리 객체들을 효율적으로 감시, 통제할 수 있게 된다. 본 논문에서는 관리자가 피관리자에게 위임할 관리 스크립트를 작성하기 위한 포괄적인 관리 알고리즘을 제안하고, OSI 장애 관리 모델을 부분적으로 구현하였다. 그리고 제안된 관리 알고리즘은 관리 정보를 병행으로 액세스할 수 있으며, 위임을 효율적으로 지원할 수 있음을 몇가지 예를 들어 설명한다. 특히 관리 알고리즘은 객체지향 병행 언어인 ABCL로 쉽게 표현될 수 있어 구현이 용이하고, 다양하고 명시적인 동기화 메카니즘을 제공할 수 있다.

ABSTRACT

Network management represents those activities which control and moitor the use of resources. Remote delegation model supports flexible and effective distributin of management functions among managers and agents, and it may cause an reliable network management in a relatively complex and high-speed networks. In this model, managers delegate to agents execution of management programs as prescribed in a management scripting language. In addition, primitives included in the management programs enable agents to monitor and control localmanaged objects effectively. We suggest management algorithms in which management scripts are delegated from managers to agents and partiallty implement OSI fault management. This mana gement algorithm can effectively support delegation and control concurrent accesses to management information. Moreover, it can be easily translated into object-based concurrent programming language : ABCL.

In this paper, we will scrutinize some essential aspects of this management.

*光云大學校 電子計算學科
Dept. of Computer Science, Kwangwoon University
**明知大學校 電子計算學科
Dept. of Computer Science, MyongJi University
論文番號 : 92-122 (接受1992. 6. 10)

Ⅰ. 서 론

대규모 네트워크는 통상적으로 분산되어 있으며, 이들은 수천 개의 각종 요소들로 구성되어 있다. 따라서 대규모 분산 시스템의 관리에는 많은 어려움을 겪게 되며, 이러한 어려움을 타개하기 위하여 네트워크 관리를 위한 표준화 작업이 진행되어 왔으나, 표준화를 위한 대부분의 노력은 관리 프로토콜과 SMI(Structure of Management Information) 모델에 집중되어 왔다.[1,2,3,4]

표준화된 모델에서 관리자와 피관리자는 표준 프로토콜을 이용하여 통신하게 되고, 이를 이용하여 다양한 관리 기능을 수행하게 된다.[1,4] 그러나 대규모의 복잡한 네트워크에서 표준화된 관리 모델과 프로토콜은 심각한 트래픽 오버헤드가 따르게 되는데 이는 네트워크 관리 정보의 교환에 기인한다. 더구나 이 프레임워에서는 모든 관리 책임이 관리자에게 묵시적으로 부여되고, 그와 같은 관리 기능의 집중화 현상은 결과적으로 많은 비용의 효율적인 관리를 초래하게 된다. 이와 같은 이유로 네트워크 관리 시스템에서의 트래픽 오버헤드를 줄이기 위한 많은 연구가 진행되어 왔으며, Internet 환경하에서 Virtual agent의 개념을 이용한 계층적 네트워크 관리 모델이 제시되기도 하였다.[5] 한편, 또 다른 시도로 관리자가 피관리자에게 관리 프로그램의 위임을 통하여 피관리측에서 직접 관리 프로그램을 수행하게 하는 위임 모델이 제시되었으며, 이 모델은 트래픽의 현저한 감소뿐만 아니라 관리자 관점에서 관리 프로그램의 설계를 가능하게 하였다.[6]

위임 모델에서는 관리 스크립트의 기술이 필요하게 되고, 이의 효율적인 기술과 위임, 병행 관리 프로그램 간의 상호작용 등을 반드시 고려해야 한다. 본 논문에서는 이러한 점을 고려하여 효율적인 관리 스크립트를 작성하기 위한 관리 알고리즘을 제시하고, 여러 가지 관점에서 이를 고찰한다. 본 논문은 먼저 2장에서 원격 위임 모델을 설명하고, 3장에서는 관리 알고리즘을 기술하기 위한 관리 기술 언어(management scripting language)로서의 ABCL(An object-Based Concurrent Language)을 소개하고, 관리 스크립트 작성을 위한 관리 알고리즘을 제시한다. 4장에서는 제시한 알고리즘에 따라 ABCL 언어로 구현한 관리 스크립트의 예를 보이고, 5장에서는 위임 모델의 설계 및 구현에 관해 살펴보았으며, 마지막으로 제시한 관리 알고리즘을 몇 가지 측면에서 분석하고 결론을 맺는다.

Ⅱ. 원격 위임 모델

2-1. 표준화 모델에서 관리자와 피관리자 간의 상호작용

분산 네트워크 시스템에서 각종 자원을 관리하기 위해서는 다양한 문제가 야기되며 ISO/IEC에서는 네트워크 관리를 위한 정보 구조와 표준 프로토콜을 권고하였으나, 관리자와 피관리자 간의 스프트웨어 구조에 대해서는 명시적으로 정의하지 않았다.[1,3]

제시된 모델에서 관리자는 M-CREATE, M-DELETE, M-SET, M-GET, M-EVENT-REPORT, M-ACTION 등의 프리미티브들을 사용하여 피관리자와 정보를 교환한다. 그림 1은 관리자, 피관리자, 관리 객체 간의 상호관계를 보여준다.

그림 1. 관리자, 피관리자, 관리 객체 간의 상호관계
Figure 1. Interaction of manager, agent and managed object

한편, 특정 사건의 발생을 감지하기 위해서 관리자는 관리자와 association의 설정시 M-EVENT-REPORT 프리미티브를 포함해야 하며, 이 프리미티브를 사용하지 않을 때는 M-GET 프리미티브를 이용하여 관련 변수를 poll해야 한다.

Polling-based 관리 기법은 SNMP(Simple Network Management Protocol)에서 사용되는 기법으로서 관리 트래픽의 대부분이 Poll에 소요되므로 관리 성능의 저하를 초래한다. 그 외에도 특정 사건의 발생을 감지하기 위해서는 짧은 시간 간격으로 피관리지를 poll해야 하는 부담이 따르게 됨은 물론, 아주 짧은 시간 간격으로 poll한다 해도 중대한 사건을 놓칠 가능성이 항상 존재하게 된다. 반면에, EVENT-REPORT 방식은 관리 트래픽을 줄일 수 있으며, CMIP(Common Management Information Protocol)에서는 M-EVENT-REPORT를 통하여 관리자에게 사건을 통지하는 방식을 채택하고 있다.[4]

한편 EVENT-REPORT 방식을 채태하고 있는 경우일지라도 두가지 이상의 조건이 동시에 발행할 때 일어나는 복합 사건(composite event)일 경우에 관리자는 사건 발생을 감지하기 위해 각각의 조건이 발생했는지 통지를 받아야 하며, 사건 통지의 지연으로

인하여 복합 사건의 감지가 불가능한 경우가 존재하게 된다. SNMP 와 CMIP에서는 이러한 기능이 미비하며, 모든 관리기능이 관리자에게만 집중되게 된다.

다음에 기술할 원격 위임 모델에서는 관리자와 피관리자 간의 불필요한 통신을 최대한 줄일 수 있으므로 위에서 열거한 문제들이 완화될 수 있다.

2-2. 위임 모델에서 관리자와 피관리자의 역할

원격 위임 모델은 모든 관리 기능이 관리자에게 집중되는 것을 방지하기 위하여 관리 기능을 관리자와 피관리자 간에 분배하는 모델로서, OSI 관리 모델의 확장된 개념으로 생각할 수 있다. 여기에서 관리 기술 언어로서 표현될 수 있는 관리 프로그램은 관리자에 의해 설계되며, 관리자는 필요에 따라 피관리자에게 수행을 위임할 수 있다. 일단 위임된 관리 프로그램은 관리자의 방해없이 피관리자에 의해 수행되며, 따라서 피관리의 구조는 표준 모델과는 다르게 표현되여야 한다.

그림2는 위임 모델에서의 관리자, 피관리자, 관리 프로그램 간의 관계를 도시한 것이다.

자가 액세스하여서는 안될 데이타는 자신의 데이타 영역(private data area)에 둠으로써 관리 프로그램에서의 통제되지 않은 액세스를 방지할 수 있다.

Ⅲ. 관리 알고리즘

앞 절에서 언급한 바와 같이 분산 환경하에서의 네트워크 관리는 복잡하며, 이들을 효율적으로 관리하기 위해서는 관리 알고리즘이 필요하게 된다. 관리 알고리즘에는 관리 대상이 되는 시스템의 동작을 관찰하고, 어떤 사건이 발생했을 때 이에 대응하는적절한 행동이 기술되어야 한다. 관리 언어는 이러한 알고리즘을 기술하고 구현하기에 적합한 수단이며 이를 위하여 관리 언어는 선언문이나 명령문 혹은 프로시져를 기술하기 위한 문장들이 제공되어야 한다. 선언문은 관리 알고리즘에서 사용되는 여러 가지의 데이타 간의 관계를 기술하기 위해 사용되며, 프로시져 기술문은 일련의 동작들을 기술하기 위한 것이고, 명령문은 즉각적으로 대응해서 수행해야 할 명령을 하기 위 한 것이다. 따라서 분산 환경에서의 관리를 위

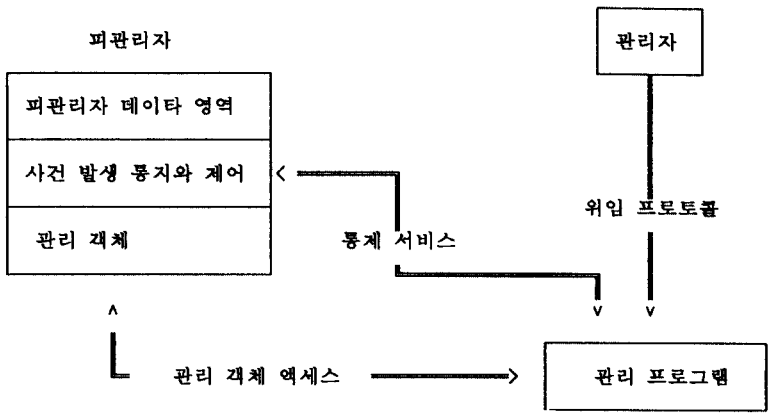


그림 2. 위임 모델에서 관리자와 피관리자의 관계
Figure 2. Interactions of manager, agent in delegation model

그림 2에서 관리자는 위임 프로토콜을 사용하여 관리 프로그램을 피관리자에게 위임하고, 위임된 관리 프로그램은 피관리자의 환경하에서 수행되게 된다. 관리 객체에 대한 액세스는 피관리자의 통제하에서 수행되며, 관리자와 피관리자 간에서는 수행의 중지나 재개 등을 위한 기법이 유지되어야 한다. 또 관리

한 언어로는 4세대 언어나 인공지능형 언어가 적절한 것으로 고려되며, 범용 고급 언어는 관리 언어가 적절한 것으로 고려되며, 범용 고급 언어는 관리 언어로는 부적절한 많은 측면을 지닌다.[7] 다음 절에서는 관리 언어로 사용하고자 하는 객체지향 병행 언어인 ABCI에 대해 기술한다.

3-1. ABCl

ABCL은 Akinori Yonezawa에 의해 개발된 언어로서, 설계 원리로는 ABCM(An Objectoriented Concurrent Computation Model)을 채택하였다.[8,9,10] ABCM은 병행 시스템이나 병행 알고리즘의 기술에 널리 이용되며, 다양한 메시지 전달 기법을 통하여 모델링 된다. 더불어 모델의 의미론(semantics)을 충분히 반영할 수 있는 다양한 버전의 언어가 개발되어 여러 분야에 이용되고 있다.[11,12,13,14]

ABCL에서 연산이나 정보처리의 단위는 객체이고, 객체는 메시지를 받으면 활성화되며, 다른 객체에 직접 액세스가 불가능하고, 메시지 교환을 통해서만 정보 교환이 가능하게 된다. 각 객체는 객체 내에서만 액세스 할 수 있는 지역 변수와 메시지가 수용될 때 수행되는 일련의 명령문들로 구성된다. 그림3은 객체의 기본 형태이다.

```
[ object object name
        ( state representation of
                                local memory...)
        ( script
          ( => message pattern where
                    constraint...    action...)
                        ...
          ( => message pattern where
                      constraint...  action...))]
```

그림 3. 기본적인 객체 정의
Figure 3. Basic object definition

그림 3에서 객체는 고유의 이름을 가지며, (state...)는 지역 변수와 그들의 초기화를 나타내고, (scipt...)부분은 메시지 수신 후 수행할 일련의 문장들이다. 객체는 메시지를 받아 활성화 된 후 다음과 같은 기본적인 작업들을 수행할 수 있다.

- 자신 혹은 다른 객체로의 메시지 전송
- 객체 생성
- 지역 변수들의 재용 참조나 갱신
- 각종 연산(산술연산, 리스트 처리 등)

또 ABCL에서 객체는 휴(dormant) 활동(active), 대기(wait)의 세 가지 상태로 표현되는데, 일단 객체가 생성되면 휴지 상태에 있게 되고 메시지를 수신하면 활동 상태로 전이된다. 활동 상태에서 해당 스크립트의 수행이 종료되면 다시 휴지 상태가 되며, 활동 상태에서 대기(wait-for) 명령어를 수행하면 대기 상태가 된다. 대기 상태에 있는 객체는 메시지 수신 후 다시 활동 상태로 전환할 수 있다. ABCL에서의 객체들은 메시지 수신에 따라 활성화되므로 다양한 메시지 전송 방법이 제공되는 것이 특징이라 할 수 있으며, 특히 병행 객체들의 동기화를 위한 편리한 수단들이 제공되고 있다. 메시지 전송은 다음과 같이 크게 세 가지 형태로 분류된다.

1)과거형 메시지 전송(past-type message passing)
2)현재형 메시지 전송(now-type message passing)
3)미래형 메시지 전송(future-type message passing)

첫째, 과거형의 메시지 전송은 메시지의 송신 후 응답에 관계 없이 다음 작업을 진행할 수 있는 현재로, 수신자에게 단순히 메시지 송신만을 하는 경우에 이용할 수 있으며, [target ⟨=message]로 표현된다.

둘째, 현재형 메시지는 메시지 송신 후에 수신자의 응답이 있을 때까지 대기하는 형태이며, [target ⟨==message]로 표현되고, 통상적인 프로그래밍에서의 함수의 프로시져 호출과 유사하나 다음 두 가지 측면에서 구분된다. 수신자가 수신한 메시지에 대한 응답을 전송한 후 함수나 프로시져는 작업을 중단하게 되나, 현재형 메시지 전송에서는 응답 송신 후에도 객체 자신의 작업을 수행할 수 있다. 또 수신한 객체가 수신 메시지에 대한 응답이 불가능할 때, 이를 응답이 가능한 다른 객체에게 위임할 수 있으며, 이러한 경우 위임받은 객체가 직접 송신 객체에게 응답을 전송한다.

셋째, 미래형 메시지 전송은 수신자로 부터의 응답은 필요하나, 즉각적으로 사용되는 전보가 아닌 경우에 이용되며, [target ⟨= message $ future-object]로 표현되고, 미래형 변수(future variable)를 설정하여 응답을 받는다. 이 때 미래형 객체는 큐와 같은 역할을 하게 되고, 수신자로 부터의 응답 메시지는 미래형 객체에 저장되며, make-future 명령어에 의해 생성된다. 송신 객체는 응답이 필요하면 미래형 객체의 내용을 조회할 수 있다. 조회를 위해 미래형 객체에 적용할 수 있는 함수는 다음과 같다.

- (ready? future-object) : 미래형 객체에 저장되어 있는 첫번째 값을 반환하는 지를 판별하기 위한 함수
- (next-value future-object) : 미래형 객체에 저장

되어 있는 첫번째 값을 반환하는 함수
(all-value future-object) : 미래형 객체에 현재 저장되어 있는 모든 요소들을 반환하는 함수

객체가 메시지 수신 후 활성화 되면 일련의 스크립트들이 수행되는데, 이 때 외부로부터 제어를 받거나 인터럽트될 수 없으며, 단지 객체 내에서만 종료될 수 있다.

그러나 긴급한 메시지가 객체에 전달되어야 할 경우가 존재하므로, 이와 같은 상황에 대비하여 ABCL에서는 긴급메시지 전송을 지원하고 있다. 이는 모델로 쉽게 표현하지 못하는 상황을 언어로서 편리하게 표현할 수 있는 대표적인 경우로서 ABCL의 큰 장점중의 하나이다. 따라서 ABCL에서 긴급 메시지는 수행중인 행위가 atomic한 경우를 제외하고는 항상 우선적으로 수락될 수 있으며, 긴급 메시니의 수행 후 재개는 resume이나 nonresume을 통해 선택적으로 지원된다. 긴급 메시지와의 구분을 위해 통상 형태의 메시지는 ordinary 메시지로 칭하고 두 가지 모드로 구분한다. 이들은 다음 형태와 같이 각각 '=〉'와 '=〉〉'를 이용하여 표현된다.

step 1 : Send some messages to agent in parallel to detect composite events. If monitoring have to performed within certain temporal tolerance then set an alarm clock.
/*복합 사건을 탐지하기 위해서는 병렬로 감시 메시지를 전송한다. 허용된 시간 내의 감시라면 타이머에게 과거형 메시지를 전송한다. 이 때 진단 결과는 미래 변수에 저장된다.*/

step 2 : On reception of express message from an agent, send future-type message to diagnostic routine.
/*피관리자로부터 긴급 형태의 사건발생 메시지를 수신하면 진단 객체에게 미래형의 진단 요구 메시지를 전송한다. 이 때 진단 결과는 미래 변수에 저장된다.*/

step 3 : Test future variable.
If ready the get a value from future-variable queue.
/*사건 발생의 진위를 가리기 위해 진단 결과가 반환되였는지를 체크한 후, 미래 변수중의 요소를 가져온다.*/

step 4 : Test whether or not condition variable is set.
If true then goto step5,
else goto step3.
/*값이 참이면 회복을 위해 step 5로 분기하고 거짓일 명령이 필요할 때마다 미래 변수를 참조하기 위해 step 3으로 분기한다*/

step 5 : Send past-type message to recovery routine.
Notify the manager of a fact.
/*회복을 위해 회복기능을 수행하는 객체에 과거형 메시지를 전송한다. 결과를 관리자에게 알린다.*/

그림 4. 관리 알고리즘
Figure 4. A management algorithm

이 알고리즘에서는 위에서 설명한 다섯 단계와 ABCL에서 제공하는 메시지 전송 기법을 명시적으로 사용할 수 있음을 보였다.

IV장에서는 2절에서 제안한 관리 알고리즘에 따라 몇가지 문제점을 해결하기 위하여 관리 알고리즘을 ABCL 언어로 구현한 예를 보이고, 구현된 알고리즘에서 지원가능한 여러 가지 해결책을 고찰한다.

Ⅳ. 관리 스크립트의 구성

관리 기술 언어는 위임된 관리 프로그램을 기술하기 위해 사용되므로 관리 프로그램의 위임이나 코디네이션(coordination) 방법을 지원해야 하고, 위임 모델에서 위임된 관리 프로그램은 병행 수행이 가능하므로 코디네이션을 조심스럽게 기술해야 한다.

부적절한 기술은 관리 행위의 정확성을 보장하기 어려우며, 특히 병행 프로그램의 비결정성(nondeterminism)에 위해 예측 불가능한 결과가 발생할 수도 있다.

본 장에서는 이러한 문제점들과 결부하여 Ⅲ장에서 제시한 관리 알고리즘에 따라 ABCL로 구성한 관리 스크립트를 제시하고, 몇 가지 측면에서 제시된 스크립트를 고찰하고자 한다.

4-1. 위 임

그림 2의 위임 모델에서 관리자는 관리 프로그램을 피관리자에게 위임한 후 위임한 프로그램의 실행과는 독립적으로 자신의 작업을 계속하게 된다. 이 때 관리자는 위임된 프로그램의 수행을 중단시키거나 재개 혹은 종료시키기 위하여 위임된 프로그램에대한 콘트롤을 유지해야 한다. 이를 ABCL로 표현할 경우 과거형 메시지 전송으로 쉽게 표현되며, 메시지 전송 후 관리자는 피관리자와 별도의 수행을 계속할 수 있다. 그리고 감시하고자 하는 것을 중단하려면 객체의 활성화를 중지시킬 수 있는 긴급 메시지를 송신하여 피관리자 객체의 수행을 중지시킬 수 있는 긴급 메시지를 송신하여 피관리자 객체의 수행을 중지시킬 수 있다. 예를 들어, 피관리자에게 링크를 감시하고 통제하도록 위임된 상황은 그림 5와 같이 ABCL을 이용하여 편리하게 기술할 수 있다.

```
[ object link
  ( state [ failure := false])
  ( script
    ( => [:monitor] from sender where
                          (= sender manager)
          [agent <==
              {
               [:watch q-length]
               [:watch control-state] ) }
  (.....
        some stuffone
  .....)
  ( => [:event-occurred] from sender where
                           (= sender agent)
        [ link-handler <<=
            [:handle-congestion failure]])
  ( if failure
    ( temporary
     [failure-params := (make-future)]
     [agent <<= [:recover failure-type] $
                           failure-params]
     [para := (next-value failure-params)]
     [manager <= [:notify para]]))]
```

그림 5. 링크를 감시하기 위한 ABCL 스크립트의 예
Figure 5. Example of a script for link failue

그림 5에서 비정상적인 조건의 탐지는 "{.....}"로 표현된 부분으로서, 현재형 메시지를 병렬로 송신하여 동시에 감시하고자 하는 객체를 활성지킨다. 따라서 복합 사건의 표현이 용이하며, 병렬로 수행된 메시지는 모든 객체로 부터의 응답이 올 때까지 대기하므로 일정한 시간내의 응답을 모두 수용할 수 있다. 따라서 ABCL로 관리 프로그램 기수리 다양한 메시지 타입과 모드를 명시적으로 기술함으로써 관리자가 모든 관리 정보를 poll하지 않고 쉽게 위임할 수 있으며, 사건 바랭의 통지는 EVENT-REPORT 형태인 긴급 메시지로 전송되므로 관리자가 대기하는 것을 막아준다.

4-2. 관리자와 피관리자 간의 통신

원격 위임 모델을 구현하기 위한 방법으로서, 이미 널리 알려진 대로 구조화된 분산 시스템에서 통신을 위한 모델들 중의 하나인 원격 프로시져 호출(Remote Procedure Call)이 있다. 이 모델에서 관리 프로그램을 관리자 측에서 작성한 경우를 고려해보자. 피관리자의 관리 스크립트 부분을 기동시키기 위해서 관리자는 원격 프로시져 호출을 수행하게 되고, 원격 호출이 일어나면 관리자는 하던 작업을 먼추고 결과가 반환될 때까지 기다리며, 결과가 반환되면 수행을 재개하게 될 것이다. 이는 원격 프로시져 호출이 관리자를 블럭시키고 관리 스크립트 부분의 수행이 완료될 때까지 대기함을 의미하는 것이다. 또, 관리 스크립트가 피관리자 입장에서 취급되므로 이는 피관리자가 사전에 관리 대상이 되는 모든 일을 예측하여 미리 관리 스크립트를 작성해야 함을 의미하는 것으로, 피관리자 설계자에게 모든 부담을 부과하게 되는 결과를 초래한다. 따라서 원격 프로시져 호출을 이용한 구현은 부적절한 해결책이라 할 수 잇다. 한편, 앞에서 제시한 바와 같이 관리자와 피관리자가 독립적으로 작업을 수행할 수 있고, 필요에 따라서만 통신할 수 있는 방법이 보다 적절한 해결책이라 사료된다.

4-3. 분산 객체들의 atomic 관리

관리 객체들은 흔히 분산되어 있으며, 이러한 분산 객체들에 대한 감시나 통제는 신중히 고려해야 한다. 분산된 객체들의 상태에 따라 발생하는 사건을 감시하는 상황을 고려해 보자. 분산된 사건은 각각 다른 시각에 발생할 수 있으며, 이러한 사건은 일련의 상태가 모여서 발생하는 것으로 고려할 수 있다, 따라서 이러한 상태와 관련하여 몇 가지 문제점들이 있을 수 있다. 즉, atomic monitoring과 atomic change이다.

Atomic monitoring은 분산된 관리 객체들이 허용된 시간 내에서 동시에 감시되는 것을 의미하고, atomic change는 대상이 되는 모든 분사된 객체들의 값이 동시에 변화되는 것을 의미한다. 이러한 문제는 전송 지연이나 대기 등의 문제고 인해 표현이 까다로운 부분이다. ABCL에서는 이렇게 동시에 메시지를 전송하는 문제는 multicast로 표현되고, 허용 시간 문제는 timer 객체를 통하여 비교적 용이하게 표현된다. 아래 예는 이러한상황을 고려한 것으로서 관리자가 과부하 상태를 감시하는 경우이다.

관리 대상이 되는 객체에 과부하가 걸릴 경우에 처리량을 줄여 주기 위한 관리 스크립트를 그림 6과 같이 ABCL로 작정할 수 있다.

```
[ object load-handler
  ( state
    [mo := nil]
    [heavy-load := (make-future)])
  ( script
    ( => [:add-mo M]
         [mo := (cons M mo)])
    ( => [:monitor :within INTERVAL] from
          sender where (= sender manager)
       [analarmclock <= [:start-and-wake
                         Me :after INTERVAL]
       [mo <= [:monitor para] $heavy-load]
         (while (not (ready? heavy-load))
                .......some stuffone .....)
    ( if heavy-load
         [mo <<= [:reduce throughput]])
                    ......)]

[object analarmclock
   (state person-to-wake count)
   (script
      ( => [:start-and-wake Person
                :after time]
           [person-to-wake := Person]
           [count := time]
           (while ( > count 0)
             (progn
                (consume-unit-time)
                [count := (subl count)]))
        [person-to-wake <<=
                         [:time-is-up]])
   (=>> [:wake Person :after time]
        (non-resume)
        [Me <= [:start-and-wake Person
                         :after time]])
   (=>> [:stop]
        (non-resume))))]
```

그림 6. atomic monitoring change를 위한 ABCL과 스크립트

Figure 6. Example of a script for atomic monitoring and change

이 예에서 과부하 상태를 탐지하려면 관련있는 모든 관리 객체들을 동시에 감시하여야 하며, 과부하가 탐지되면 관련되는 모든 객체에게 동시에 처리량을 줄일 것을 명령해야 한다. 따라서 각 관련 객체들을 리스트로구성하였으며, atomic monitoring과 atomic change는 이 리스트에 메시지를 전송하는 multicast를 이용하여 표현하였다. 한편 시간적인 여유를 표현하기 위해서는 별도의 timer 객체인 analarmclock을 사용하여 시간 허용치를 초과하면 긴급 메시지를 전송하여수행을 중단시키도록 표현하였다.

V. 구 현

본 논문에서 설명한 위임 모델을 구현하기 위하여 IOS에서 권고한 관리 기능 중의 하나인 OSI 장애 관리를 선택하였으며, 그림 7은 장애 관리를 위한 관리자와 피관리자 구현 모델이다.

위임 모델의 구현을 위해 SDT-200 SPARC 워크스테이션에서 OSI 상위 계층은 ISODE6.0을 이용하고, 하위 계층으로 TCP/IP를 이용하여 관리자와 피관리자 간에 장애 정보를 교환하도록 하였다. 관리지와 피관리자 간의 연결(association)을 설정하거나 해제하기 위해서는 ISODE 의 ACSE 서비스를 사용하여 구현하였고, 한편 IV장에서 제시한 관리 스크립트의 구성을 위해서는 SUN4 시스템에서 수행될 수

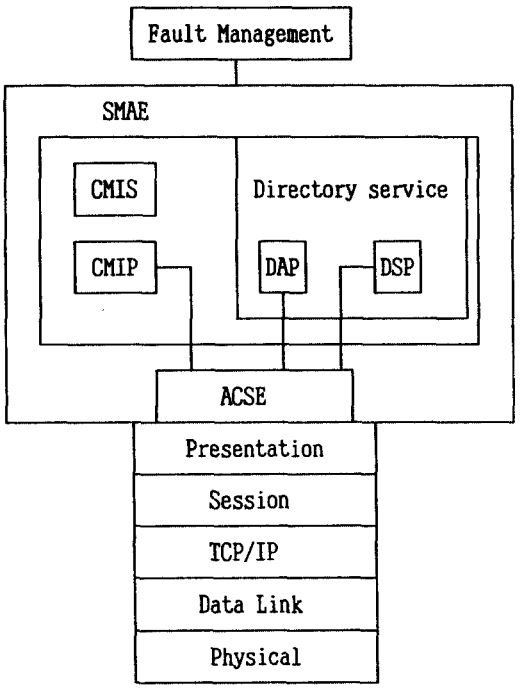


그림 7. OSI 장애 관리를 위한 관리 모델
Figure 7. Management model for fault management

있도록 Y.Ichisugi가 구현한 ABCL/R KCL(Kyoto Common Lisp)version을 이용하였다. 장애 관리를 위해 필요한 객체 관리, 상태 관리, 경보 보고, 사건 보고 관리, 로그 제어관리 등의 시스템 관리 기능 중 정보 보고 서비스를 위주로 구현하였으며, CMISE서비스 사용자가 경보를 보고하기 위해서는 통보서비스의 하나인 M-EVENT-REPORT 서비스를 이용하였다. 경보 보고를 위한 서비스 파리미티브 중 커널 기능 단위인 경보서비스를 중심으로 피관리자 시스템 재의 통신, 서비스 품질, 처리, 장치와 환경에 관한 경보를 관리자에게 보고하도록 하였다. 경보 보고 서비스를 위한 파라미터의 CMIS 파라미터로의 사상은 표1과 같다.

표 1. 커널 기능 단위 파라미터의 CMIS 파라미터 사상
Table 1. Mapping for kernel function unit

경보보고 파라미터	CMIS 파라미터
Invoke identifier	Invoke identifier
Mode	Mode
Managed object class	Managed object class
Managed object instance	Managed object instance
Alarm type	Event type
Event time	Event time
Probable cause	Event argument
Specific problem	Event argument
Backup status	Event argument
Backup object instance	Event argument
Trend indication	Event argument
Triggered threshold	Event argument
Threshold level	Event argument
Observed value	Event argument
Notification identifier	Event argument
Correlated notifications	Event argument
Monitered attributes	Event argument
Proposed repair action	Event argument
Problem text	Event argument
Problem data	Event argument
Current time	Current time
Event result	Event result
Errors	Errors

위 파라미터의 의미중 몇 개를 살펴보면 Mode는 경보 보고가 확인형인지 비확인형인지를 나타내며, Backup status는 장애가 발생한 객체가 여분으로 보관되어 있는지에 대한 상태를 표시한다. 또 Alarm type은 경보형을 분류하기 위한 것으로 5가지 형으로 나누어지며 프로시저나 프로세스와 관련된 경보형인 통신 경보형(Communications alarm type), 서비스 품질과 관련된 서비스 경보형(Quality of service alarm type), 소프트웨어의 오류로 인한 처리 경보형(Processing alarm type), 장비의 고장과 연관된 경보형인 장비 경보형(Equipment alarm type)과 장비를 둘러싸고 있는 환경과 관련되는 조건에 관한 경보형인 환경 경보형으로 나누어진다. 그 외 Probable cause는 경보가 발생가능한 이유에 대한 정보를 제공하는 것으로, 각 경보형에 대한 발생가능한 원인이 등록되어 있다. 예를 들면 환경 경보형의 발생원인으로서는 연기를 감지한다거나, 습도가 높다든가 하는 이유 등이 있으며, 통신 경보형의 발생 원인으로 감지한다거나, 습도가 높다든가 하는 이유 등이 있으며, 통신 경보형의 발생 원인으로서는 시그널을 잃거나, 프레임 오류, 지역정보전송 오류 등이 있다. 그리고 Correlated notivication은 상호 관련이 있는 통지가 존재하는지의 여부를 알려준다.

구현을 위한 서비스 프리미티브는 ISO/IEC 표준에 따라 파라미터를 설정하여 함수로 구현하였고, 파라미터의 설정을 위해서는 INTAP(Interoperability Technology Association for Information Processing)에서 제시한 OSI Management Implementation Specificaton을 참조하였다.

Ⅵ. 결 론

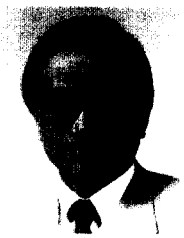
실시간 처리와 분산 환경에서의 처리 문제는 네트워크 관리의 중요한 과제로 부가되고 있으며, 이러한 점을 고려한 원격 위임 모델은 관리자와 피관리자 간에 관리 기능을 필요에 따라 분배하는 OSI 모델의 확장된 개념으로 볼 수 있다. 또 이를 지원하기 위한 관리 언어는 분산 환경에서 관리 알고리즘을 구현하기 위한 적절한 방법이다.

본 논문에서는 객체치향 병행 언어인 ABCL이 분산 환경에서 객체의 atomic monitoring과 atomic change를 편리하게 지원할 뿐만 아니라 원격 위임 모델에서의 위임도 쉽게 표현할 수 있음을 살표보았다. 또 이를 이용한 관리 알고리즘을 제시하고, 이 알고리즘이 관리 스크립트를 구성하기에 편리하고, 위임을 통하여 피관리자에게 관리 기능을 적절하게 분배함으로써, 피관리자 측면이 아닌 관리자 측면에서 관리 스크립트를 표현할 수 있도록 하였다. 특히, 명시적인 동기화 기법을 제공하는 것이 특징이라 할 수 있고, 위임 모델의 구현을 위해 OSI 장애 관리 중 경보 보고 서비스를 선택하여 이를 부분적으로 구현하였다. 한편 위임 모델의 정확한 성능을 평가하기 위해서는 OSI 계층 구조에서 성능 평가를 위한 파라미터들을 어떻게 설정하고 어디서 얻는가에 따라 결과치가 다를 수 있으므로, 파라미터의 선정에 신중을 기하여야 할 것이다. 앞으로 위임 모델의 성능 평가를 위한 성능 관리 모델과 성능 평가 도구가 연구되어야 할 것이며, 다른 모델들과의 정량적인 비교분석이 이루어져야 할 것이다. 이와 더불어 효율적인 관리 스크립트 작성에 대한 지속적인 연구와, 위임 프로토콜에대한 연구도 뒤따라야 할 것이다.

참 고 문 헌

1. ISO/IEC DIS10040, Systems Management Overview, 1990
2. OSI/Network Management forum Architecture, Forum 004, issuel, January 1990.
3. ISO/IEC DIS 10165-1, Structurue of Mangement Information -Part 1:Management Information Model.
4. ISO/IEC IS 9596-2, Mangement Information protocol specification-Part 2:Common Management iInformation Protocol, 1989.
5. Yuka Kamizuru, et al., "A Proposal for a Network Management Model on the Widely Intergrated Distributed Environment, Proceedings of 6th International Joint Workshop on Computer Communications, 1991, pp.319-324.
6. Yechiam Yemini, et al., "Network management by delegation, " Proceedings of the IEIP TC6/WG6.6 Second International Symposium on Integrated Network Management, 1001, 95-107.
7. David Holden, "Predictive languages for management," Proceedings of the IFIP TC6/WG6.6 International Symposium on Integrated Network Management, 1989, pp.585-596.
8. Akinori Yonezawa, "ABCL An Object-Oriented Concurrent System," The MIT Press 1990,.
9. J.P.Briot and A.Yonezawa, "Inheritance and Synchronization in Concurrent OOP.," Proceedings of ECoop, 1987, pp.35-43.
10. Etsuya Shibayama and Yuuji Ichsugi, "The ABCL/1 user guide," The MIT Press, 1990.
11. T.Watanabe and A.Yonezawa, "Reflection in an object-oriented Concurrent language," In proceedings of ACM Comference on Object-Oriented Programming Systems, Languages and Applications, San Diego CA., 1988, pp. 306-315.
12. N.Doi, Y.Kodama, and K.Hirose, "An Implementation of an Operating System Kernel using Concurrent Object-Oriented Language ABCL/c+,"Lecture Notes in Computer Science, Vol.322, Springer-Verlag, 1988, pp. 250-266.
13. Toshihiro Takada and Akinori Yonezawa, "An Implementation of an Object-Oriented Concurrent Programming Language in Distributed Environments,"The MIT Press, 1990, pp. 133-156.
14. Satoshi Matsuoka, Takuo Watanabe, Akinori Yonezawa, "Hybrid Group Reflective Architecture for Object-Oriented Concurrent Reflective Programming," Lecture Notes in Computer Science, Vol.512, Springer-Verlag, 1991, pp.231-250.

韓 順 姬(Soon Hee Han) 正會員
1960年 8月 9日生
1983年 2月 : 慶北大學校 電子工學科 學士
1985年 2月 : 光云大學校 大學院 電子計算學科 碩士
現在 : 光云大學校 大學院 電子計算學科 博士課程 修了
1989年~現在 : 國立麗水水產大學 컴퓨터工學科 助教授



李 基 鉉(Kee-Hyon Lee) 正會員
1941年 12月 25日生
1965年 2月 : 成均館大學校 經濟學科 卒業(學士)
1972年 2月 : 成均館大學校 經營大學院 情報處理 專功 (經營學碩士)
1986年 8月 : 光云大學校 大學院 電子計算學科 卒業(理學碩士)
1989年 8月 : 光云大學校 大學院 電子計算學科 博士課程 修了
1972年 5月~1976年 2月 : 총무처 행정전산 계획실 전산처리관
1976年 2月~1982年 3月 : 대한손해보험협회 전산실장
1982年 4月~現在 : 明知大學校 工科大學 電子計算學科 副教授, 明知大學校 電子計算所長
※관심분야 : 컴퓨터 네트워크(특히, OSI 관리), S/W공학

趙 國 鉉(Kuk-Hyun Cho) 正會員
1953年 12月 22日生
1970年 3月~1977年 2月 : 漢陽大學校 電子工學科 卒業(工學士)
1979年 4月~1981年 3月 : 日本 東北大學 大學院 電子通信工學科 卒業(工學碩士)
1981年 4月~1984年 3月 : 同 大學 大學院 同 學科(工學博士)
1984年 3月~現在 : 光云大學校 電子計算學科 助教授, 副教授
1987年 4月~1991年 2月 : 한국정보과학회 정보통신분과위원회 총무, 운영위원
1991年 3月~現在 : 同 分科委員長
※관심분야 : 컴퓨터 네트워크, 프로토콜 성능평가