CIM/WBEM 표준 기반 시스템 자원 관리 요소 기술 분석

An Analysis of System Resource Management Technologies Based on CIM/WBEM Standards

김지연(J.Y. Kim) 시스템소프트웨어연구팀 연구원
안창원(C.W. Ahn) 시스템소프트웨어연구팀 선임연구원
김영호(Y.H. Kim) 시스템소프트웨어연구팀 연구원
조희남(H.N. Cho) 시스템소프트웨어연구팀 연구원
정성인(S.I. Jung) 시스템소프트웨어연구팀 선임연구원, 팀장

분산 네트워크상에서는 수 많은 종류의 시스템 자원이 연결되어 있고 이들 자원을 효과적으로 관리하기 위한 시스템 자원 관리 솔루션이 제공되고 있다. 이들 솔루션은 업체마다 독자적인 개별 기술로 구현되어 시스템 자원 관리 솔루션 간 상호호환성이 보장되지 않기 때문에 솔루션 구매자는 솔루션 통합과 유지 보수에 소요되는 비용을 감당하기 힘들었다. 이러한 문제를 해소하기 위하여 DMTF는 선도적인 기업체를 중심으로 학계와 협력하여 분산 시스템 자원 관리에 대한 표준 규격을 정의 개발해 나가고 있다. 본 논문에서는 DMTF에서 제시하고 있는 자원 관리 정보를 기술하는 CIM 명세와 스키마, 이를 적용하여 구현된 웹 기반 시스템 자원 관리 아키텍처인 WBEM을 소개하고, WBEM에서 적용되는 기술인 CIMOM, CIM-XML, 프로바이더, 저장소 관련 기술에 대해 설명한다.

I. 서 론

분산 네트워크상에서는 수 많은 종류의 컴퓨터와 장비들이 연결되어 있다. 이들을 효과적으로 관리하기 위해서는 장비 제공 업체들이나 관리 소프트웨어 업체들이 제공하는 관리 도구들을 사용하여야 한다. 이들 대부분은 자사 장비에서 적용되는 기술을 이용하여 시스템 관리 솔루션을 개발해 왔기 때문에 이종 시스템 솔루션 간 상호호환성이 보장되지 않는다. 이와 같이 상호호환성이 보장되지 않기 때문에 솔루션 통합 및 유지에 많은 비용이 발생하고 있으며, 이로 인한 총 소유비용(TCO)은 급격하게 증가하고 있다.

이러한 비효율적인 상황을 타개하기 위해서는 솔루션 간 상호호환성을 제고하기 위한 시스템 자원 관리 표준 규격이 요구되고 있으며, 이에 따라 선도적인 기업체와 교육 기관이 협력하여 분산 시스템 자원에 대한 표준 규격을 정의하고 있다. DMTF (Distributed Management Task Force)는 세계적인 IT 업체들이 모여 시스템 관리 솔루션의 상호 호환성을 제고하기 위하여 관리 정보 표준 모델인 CIM(Common Information Model)을 정의해 나가고 있으며, CIM을 기반으로 시스템 자원 관리 프레임워크를 구현한 WBEM(Web Based Enterprise Management)를 이끌어 나가고 있다.

CIM은 관리 대상 자원을 다루기 위하여 객체로 정의한 일련의 정보 체계를 의미한다. 명세와 스키마로 구성되며 명세는 스키마에서 도식화된 정보를 설명하고 각 정보를 표현하기 위한 문법과 각종 타입, 그리고 이들을 표현하는 방법을 기술한다. 스키마는 관리 정보를 UML(Unified Modeling Language)을 이용하여 계층적으로 도식화한다.

WBEM은 CIM을 기반으로 분산 네트워크 환경에

서 효율적인 시스템 관리 프레임워크를 제시한다. DMTF는 WBEM 구현물을 통해 시스템 관리 프레임워크 구조를 증명하고 각 컴포넌트 내에 적용된 기술을 검증한다. WEBM은 크게 클라이언트, CIM 서버, 프로바이더(provider)로 구성된다. WBEM 구현물에는 CIM-XML 인터페이스 표준, CIM이 적용된다.

본 논문에서는 DMTF가 제시하는 시스템 관리 표준인 CIM/WBEM 기술을 소개하고 이를 구성하는 요소 기술을 중점적으로 설명한다. CIM의 개념과 이를 애플리케이션에 적용하는 방법을 설명하고, CIM을 기반으로 구현된 WBEM 구현물의 구조를 설명한다. 그리고 시스템 관리 표준 프레임워크의 내부 컴포넌트 기능과 특성을 다루고, 마지막으로 시스템 자원 관리 영역에서 DMTF를 중심으로 한 표준화 동향을 소개한다.

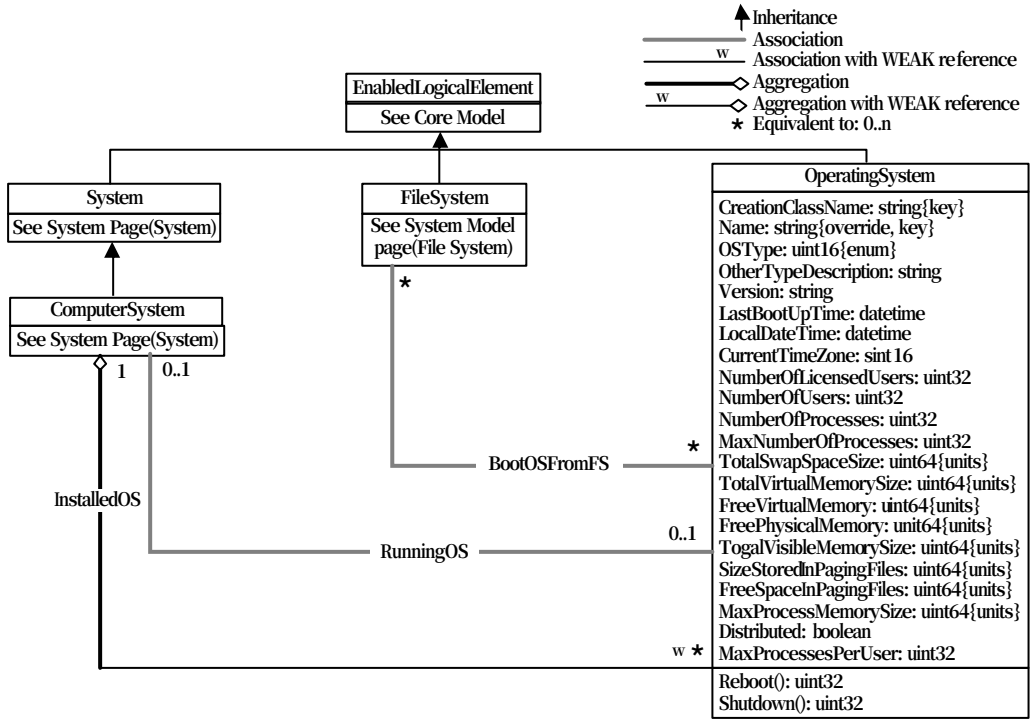
II. 시스템 자원 관리 기술 표준

1. CIM

CIM은 시스템 자원 관리 정보를 기술하는 것으로, 구현에 독립적인 스키마(schema)를 제공하는 공통 정보 모델이다. 즉, 객체 지향 모델링을 이용해 관리 대상 시스템과 망 자원을 기술하는 구조적이며 개념적인 수단을 제공한다. 객체 지향적 모델을 표현하기 위해서는 UML을 이용한다. UML은 OMG (Object Management Group)에서 개발한 객체 지향 모델링 언어로, 시스템 구성요소와 그들 간의 관계에 대한 개념을 다이어그램으로 표현하는 그래픽 언어이다[1].

CIM은 CIM 명세(specification)와 스키마로 구성된다. 현재 정의되어 공포된 CIM 명세의 버전은 2.2이고 스키마의 버전은 2.8.1이다. CIM 명세는 CIM 스키마를 기술하는 메타 스키마(meta schema)를 정의하고, 메타 스키마를 MOF(Managed Object Format)로 표현하기 위한 메타 스키마 구성요소들의 MOF 컴포넌트 표현에 대해 기술한다.

CIM 스키마는 관리 정보를 속성(property)과 관계(association)를 가진 클래스로 표현하고, 이를 UML로 도식화 함으로써 프로그래머가 쉽게 이해할 수 있고 스키마 디자이너가 쉽게 확장 가능하도록



(그림 1) CIM 스키마

작성되어 있다((그림 1) 참조).

CIM 스키마는 핵심 모델(core model), 공통 모델(common model), 확장 스키마(extension schema) 등 3가지 개념적인 계층으로 이루어져 있다[2].

• 핵심 모델

핵심 모델은 관리 대상이 되는 정보 중 가장 기본이 되며 모든 관리 대상에 적용할 수 정보 모델로서, 공통 모델과 확장 스키마로 확장할 수 있는 기반이 된다.

• 공통 모델

공통 모델은 특정 관리 분야에 적용될 수 있는 정보 모델을 표현하지만 특화된 기술 또는 구현에 의존적 정보 모델을 의미하지는 않는다. 공통 모델은 Application, Database, Device, Event, Interop, IPsecPolicy, Metrics, Network, Physical, Policy, Support, System, User 등으로 구분되어 있다.

모델을 구성하는 클래스, 프로퍼티, 연관관계, 그리고 메소드는 프로그램 디자인이나 구현을 하기에 충분할 정도의 정보를 제공한다.

• 확장 스키마

확장스키마는 공통 모델을 확장시켜 기술 특화된 정보를 표현하는 데 사용된다. 이러한 스키마들은 운영체제와 같은 환경에 종속적이다.

DMTF에서는 스키마를 구조적으로 도식화하기 위해 UML을 사용하고 이를 실제 구현에 적용하기 위해 표현하는 방법으로 MOF라는 인터페이스 정의 언어(Interface Definition Language: IDL)와 XML (Extensible Markup Language)을 권고한다.

MOF는 UML로 도식화된 관리 스키마를 클래스, 관계(Association, Aggregation, Inheritance), 속성, 메소드 4개 요소를 이용하여 텍스트 형태로 기술하는 언어이다.

(그림 2)는 (그림 1)의 스키마를 MOF로 표현한 것이다. MOF는 클래스간의 구조와 관계, 프로퍼티의 특성을 표현하기에 적당하고 가독성이 높은 언어이다. 스키마는 XML로도 표현될 수 있으나 가독성

```
Class CIM_OperatingSystem : CIM_EnabledLogicalElement{

...
   [Description ("Time when the OperatingSystem was last booted."),
      MappingStrings {"MIF.DMTF|General Information|001.5"}]
   datetime LastBootUpTime;

   [Description (
      "OperatingSystem's notion of the local date and time of day."),
      MappingStrings {"MIB.IETF|HOST-RESOURCES-MIB.hrSystemData",
         "MIF.DMTF|General Information|001.6"}]
   datetime LocalDateTime;

   [Description (
      "Number of process contexts currently loaded or running on"
      "the OperatingSystem."), Gauge,
      MappingStrings {"MIF.DMTF|Host System|001.5",
        "MIB.IETF|HOST-RESOURCES-MIB.hrSystemProcesses"}]
   uint32 NumberOfProcesses;

   [Description (
      "Number of user licenses for the OperatingSystem."
      "If unlimited, enter 0.")]
   uint32 NumberOfLicensedUsers;

*** * *
```

(그림 2) MOF로 표현된 스키마

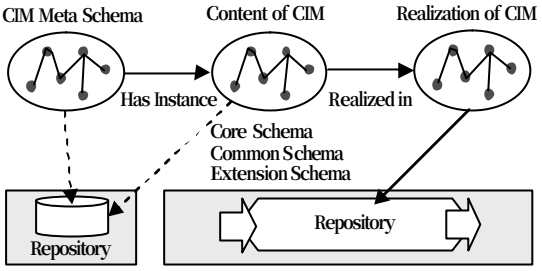
이 낮고 스키마를 XML로 변환한 경우 텍스트 데이터 크기가 크다. 하지만 XML은 구현 수준에서 가독성이 높은 언어이므로 MOF는 프로바이더가 등록될 때 사용되고 그 이후로는 프로그램이 이해하기 쉬운 형태인 XML을 이용하는 경우가 일반적이다[3].

(그림 3)은 CIM이 정의되는 단계부터 구현에 이르는 과정까지 변환되는 과정을 도식화한 것이다.

CIM 메타 스키마는 CIM 스키마를 정의하는 데 사용되는 일종의 문법이다. CIM 메타 스키마는 Class, Method, Qualifier, Association, Reference, Indication이 무엇인지 정의하고 이들 간의 관계 정립을 위해 요구된다. 그러므로 CIM 메타 스키마는 관리 대상 정보에 대한 속성과 관계를 포함하지 않는다.

CIM 메타 스키마를 이용하여 관리 대상 정보를 실제적으로 표현한 것이 핵심 모델, 공통 모델, 그리고 확장 스키마이다.

애플리케이션은 CIM에서 정의한 정보 모델을 적용하고자 할 때 핵심 모델, 공통 모델, 그리고 확장 스키마를 그대로 이용하지 못한다. 앞서 언급한 바와 같이 핵심 모델, 공통 모델, 그리고 확장 스키마는 UML로 도식화되어 있으며, 이를 애플리케이션에 적용하기 위해서는 애플리케이션이 이해하는 형



(그림 3) CIM 스키마의 형상화 과정

태로 변환해야 한다. 일반적으로 WBEM 구현물에서는 XML로 변환하여 사용한다.

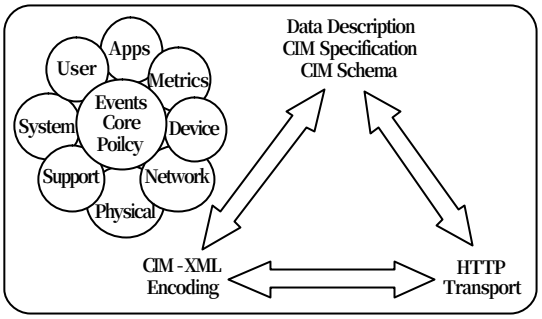
2. WBEM

DMTF에서는 분산 네트워크 환경에서 다양한 시스템 자원을 일관된 방법으로 관리하기 위한 관리 구조에 대한 지침을 표준으로 제시한다. DMTF에서는 분산 네트워크 환경에서 요구되는 다양한 시스템 자원 관리 기술을 여러 가지 워킹그룹을 구성하여 기술과 표준화에 대한 논의를 진행하고 있다. 이들 워킹그룹 중 SMWG(Server Management Working Group)는 시스템 관리 구조와 관리 대상 정보, 관리 구조를 구성하는 컴포넌트간의 프로토콜 등에 관한 기술적 논의를 진행해 나가고 있다. WBEM은 시스템 관리에 있어서 핵심이 되는 기술이다.

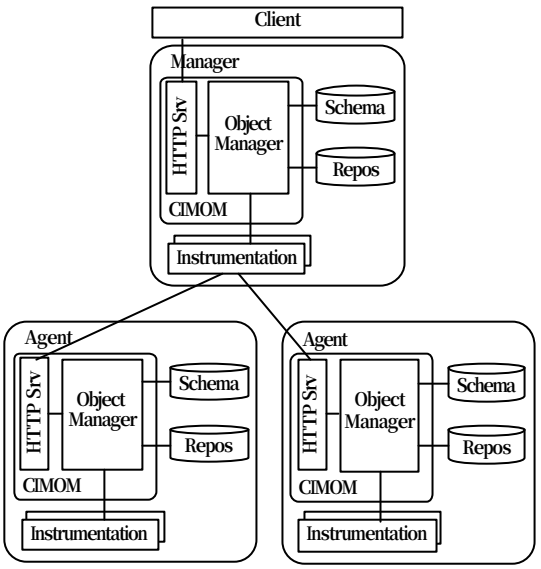
WBEM은 (그림 4)와 같이 관리 데이터 모델인 CIM 표준 정보 모델, 데이터 전송을 위해 사용되는 프로토콜과 CIM 데이터를 인코딩하는 표준 기술로 구성된다. 현재 전송 프로토콜로는 HTTP, CIM 데이터의 인코딩에는 XML을 사용하고 있다.

WBEM 시스템을 구성하기 위해서는 WBEM 클라이언트와 적어도 하나 이상의 WBEM 서버가 있어야 한다. WBEM 서버의 구성 요소는 다른 관리 시스템에서는 에이전트라고 불리는 다수의 WBEM 프로바이더와 CIM 서버인 CIMOM(CIM Object Manager), 그리고 WBEM 클라이언트와의 통신을 위한 인터페이스로 구성된다. 일반적으로 CIM 서버와 CIMOM은 동일체로 간주되고 있다.

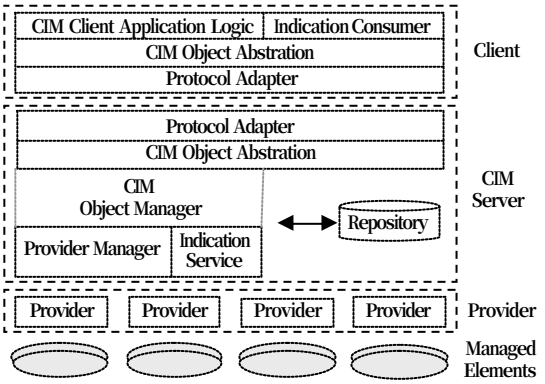
(그림 5)는 WBEM 기술 기반 시스템 관리 구조를 보여주고 있다. WBEM 구조에서는 관리자의 역



(그림 4) WBEM 표준 기술 관계도



(그림 5) WBEM 시스템 관리 구조



(그림 6) 시스템 관리 구조

할이 특정 노드에 한정되지 않고 모든 노드가 관리자의 역할을 분산 수행할 수 있다.

따라서 관리 작업은 관리자뿐만 아니라 관리 에

이전트에서도 수행된다. 즉, WBEM 구조는 관리 기능을 분산시킬 수 있으며, 관리 객체를 모델링 함으로써 객체에 대한 접근을 쉽게 해준다. 그러나 여전히 관리자의 명령에 의하여 이루어지므로 자율적인 관리를 위해서는 에이전트의 지능이 요구된다.

WBEM 기술 기반 관리 구조에서 한 시스템의 내부 구조를 상세히 도식화하면 (그림 6)과 같다. 시스템은 크게 3가지 컴포넌트로 구성된다[4].

가. 클라이언트

클라이언트는 DMTF의 "CIM Operation over HTTP spec 1.0"에 따라 서버와의 인터페이스를 구현하며, HTTP 프로토콜을 사용하여 서버 사이에 요청(request) 메시지와 응답(response) 메시지를 주고 받게 된다.

클라이언트와 서버 사이에 적용되는 보안 정책은 Authentication, No Auth, Digest Auth, SSL, SHTTP 등의 5 가지를 사용할 수 있는데, 이들 중 어떤 보안 정책을 적용하는가는 구현 주체자인 개발자에 따라 달라질 수 있다.

(그림 7)은 클라이언트가 CIM 서버로 송신하는 실제 메시지를 보여준다. HTTP 헤더에는 오퍼레이션의 종류와 각 파라미터가 기술되어 있고, HTTP 바디에는 XML로 인코딩된 실제 데이터가 기술되어 있다.

HTTP Header

```
M-POST/cimom HTTP/1.1
HOST: www.erewhon.com
Content-Type: application/xml; charset={"utf-8"
Content-Length: xxx
Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: GetClass
73-CIMObject: root/cimv2
```

HTTP Body

```
<?xml version= "1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
 <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
  <SIMPLEREQ>
   <IMETHODCALL NAME="GetClass">
    <LOCALNAMESPACEPATH>
     <NAMESPACE NAME="root"/>
     <NAMESPACE NAME="cimv20"/>
    </LOCALNAMESPACEPATH>
    .....
   </IMETHODCALL>
  </SIMPLEREQ>
 </MESSAGE>
</CIM>
```

(그림 7) CIM 오퍼레이션 요청 메시지

<표 1> CIM Operations

Functional Group	Dependency	Methods
Basic Read	None	GetClass EnumerateClasses EnumerateClassNames GetInstance EnumerateInstances EnumerateInstanceNames GetProperty
Basic Write	Basic Read	SetProperty
Instance Manipulation	Basic Write	CreateInstance ModifyInstance DeleteInstance
Schema Manipulation	Instance Manipulation	CreateClass ModifyClass DeleteClass
Association Traversal	Basic Read	Associators AssociatorNames References ReferenceNames
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier SetQualifier DeleteQualifer EnumerateQualifiers

DMTF에서 정의한 "CIM Operation over HTTP spec 1.0"에서 규정하는 오퍼레이션 메소드는 <표 1>과 같다[5].

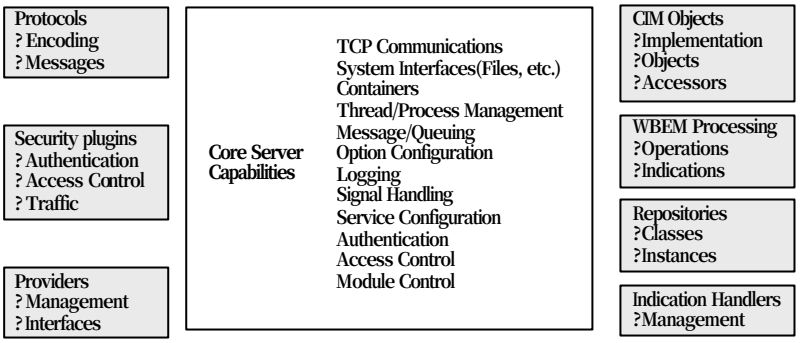
나. CIM 서버

클라이언트로부터 수신되는 요청을 처리하는 서버 프로그램을 CIM 서버라 한다.

(그림 8)은 CIM 서버가 갖추어야 할 기능을 나타내며, 그 기능을 나열하면 다음과 같다[6].

- 프로토콜 프로세싱(XML encoder/decoder)
- 보안 관리
- 프로바이더 관리
- 인디케이션 관리
- 저장소(repository) 관리
- CIMOM

CIM 서버는 작업을 관할하는 서버이므로 클라이언트로부터 작업 요청을 수신하는 모듈을 가진다. 클라이언트로부터 모든 작업 요청을 수신하기 전에 클라이언트와 서버는 인증 과정을 거친다. 인증 과



(그림 8) CIM 서버의 기능

정을 거친 클라이언트의 요청은 CIMOM으로 전달된다. CIM 서버로 수신되는 작업 오퍼레이션은 DMTF의 "CIM Operation over HTTP 1.0" 명세를 따른다. 오퍼레이션은 HTTP 프로토콜의 바디에 XML 형식으로 담겨진다. CIMOM은 XML 파서를 이용하여 오퍼레이션을 해석하고 해당되는 프로바이더로 작업을 요청한다. 프로바이더로부터 작업 결과를 전달 받아 XML로 변환하고 HTTP 프로토콜을 이용하여 클라이언트로 전송한다.

1) CIMOM

CIM 서버의 핵심 요소들 중 하나는 CIMOM이지만 일반적으로는 CIM 서버와 CIMOM은 동일체로 간주되고 있다. CIMOM은 WBEM 시스템의 CIM 객체들을 관리하는 소프트웨어이다. CIM 객체는 자바 클래스 형태로 내부에 저장이 되며, WBEM 클라이언트와 CIMOM 저장소, 그리고 관리 자원들 사이에 정보를 전송하는 역할 등을 수행한다. 프로바이더는 일반적으로 관리되는 시스템 자원별로 구현되므로 CIMOM은 많은 수의 프로바이더를 관리해야 한다. 또한 다수의 CIM 오퍼레이션에 대해 대응해야 하므로 CIMOM에는 큐잉, 스레드, 시그널 핸들링과 같은 기술이 기본적으로 요구된다.

CIMOM은 프로바이더가 CIMOM에 등록될 때 MOF로 저장된 CIM 스키마를 프로그램이 이해할 수 있는 형태로 변환해야 한다. WBEM 구현물에 따라 자바 언어에 의존적인 클래스나, 바이너리, XML과 같은 파일로 변환하여 저장소에 저장한다.

CIMOM은 프로바이더를 관리하는 프로바이더 관리 기능을 지원한다. 프로바이더 관리자는 프로바이더의 등록, 구동, 정지, 제거 등의 기능을 지원한다. 다양한 언어로 구현된 프로바이더를 지원하기 위해 프로바이더 관리자는 CIMOM과 프로바이더 간의 추상계층을 제공한다.

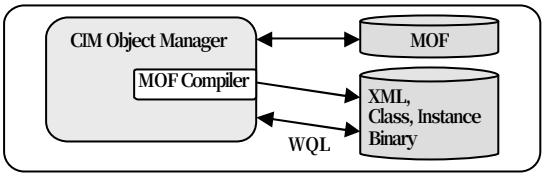
모니터링이 요구되는 프로바이더는 인디케이션 메커니즘을 이용한다. 인디케이션이란 이벤트와 같은 의미로서, 주기적인 모니터링 작업이 요구되는 시스템 자원 정보에 대하여 사용한다. 인디케이션 메커니즘을 이용하여 통지하고 이를 클라이언트에게 통보한다.

2) 저장소

저장소는 프로바이더를 통해 추출된 정보를 저장하고 검색하는 데 사용된다.

CIM 스키마는 객체 관리 그룹(OMG)에서 정의한 UML, CIM 사양서에서 정의된 MOF, 그리고 백서와 XML 등 4가지 형태로 표현할 수 있다. UML은 다이어그램으로 표현되어 사람이 이해하기 쉽도록 도움을 준다. MOF 또한 UML의 특성을 그대로 반영해 텍스트를 이용하여 구조적으로 표현하여 사람이 이해할 수 있도록 도움을 준다.

애플리케이션이 CIM 모델을 해석할 때 MOF를 사용할 수 있으나, MOF는 CIM 모델을 구조적으로 표현하기 위한 형식일 뿐 애플리케이션은 MOF를 이용하지 않는다. 애플리케이션이 다루는 모든 정보를 MOF로 변환하는 것은 초기화 단계에서 CIM 모델을 해석할 때 발생한다. CIMOM은 프로바이더가 추출하는 정보를 MOF로 변경할 필요는 없다.



(그림 9) MOF와 XML의 관계

(그림 9)는 일반적인 WBEM 구현물의 저장소를 보여준다. 일반적인 WBEM 구현물은 MOF를 가진 스키마 저장소와 MOF 컴파일러로 컴파일된 애플리케이션이 이해할 수 있는 형태의 정보를 저장하는 저장소가 구분되어 있다.

WBEM의 저장소는 네임스페이스로 접근할 수 있다. 네임스페이스는 하나 이상의 스키마들을 저장시키는 디렉토리와 같은 구조의 공간을 일컫는다. 네임스페이스는 또 다른 네임스페이스 및 class, instance, qualifier 타입들을 포함하며, 이 이름공간 안에서의 모든 객체들의 이름은 유일하다. 파일 시스템으로 저장소를 구현한 OpenPegasus는 다음과 같은 세 개의 디렉토리에 프로바이더 정보가 저장된다.

- repository/root#cimv2/classes
- repository/root#cimv2/instances
- repository/root#cimv2/qualifiers

저장소를 구성하는 방법은 구현 방법에 따라 파일 시스템과 데이터베이스를 이용할 수 있다. 데이터베이스는 XML 전용 데이터베이스 또는 관계형 데이터베이스를 이용할 수 있으며, 관리 대상 정보를 데이터베이스로 관리하는 경우 CIM 클래스는 Table로, CIM 인스턴스는 Row로, CIM 프로퍼티는 Column으로 구현된다. 질의 및 검색을 위해 WQL (WBEM Query Language)을 사용한다.

다. 프로바이더

프로바이더는 CIMOM과 하나 또는 그 이상의 관리 자원 사이의 중개자와 같은 역할을 한다. 프로바이더는 CIM 서버로부터 요청을 전달받아 해당 시스템 관리 자원에게 전달하고 관리 자원으로부터 얻은 데이터를 CIMOM에게 전달한다. 프로바이더가 구동되기 위해서는 CIMOM에 등록되어야 한다. 등록 과정에 CIMOM은 프로바이더의 관리 대상 정보를 저장소에 생성한다. CIM 서버는 클라이언트로부터 특정 정보에 대한 요청을 받으면 해당 프로바이더를 구동시킨다. 다수의 다양한 프로바이더를 관리하는 프로바이더 관리자가 CIM 서버에 구현되어 있다.

CIM 서버와 프로바이더 간 프로그래밍 인터페이스로는 CMPI(Common Manageability Programming Interface)와 NPI(Native Provider Interface)가 있다[7]. CMPI는 C언어로 NPI는 자바로 구현되어 있다. 두 개의 인터페이스 모두 유사한 기능을 제공하지만 CMPI가 최근 더 많이 사용되고 있다. 또한 CMPI는 WBEM 이니셔티브에 의해 표준화가 진행되고 있다.

3. 시나리오

(그림 10)은 CIM 서버를 통해 WBEM 구조의 각 컴포넌트가 동작하는 기본적인 흐름을 보여주고 있다.

가. 프로바이더 등록 - ①

- 프로바이더는 CIMOM에 자신을 등록한다.
- CIMOM은 MOF로 기술되어 있는 해당 프로바이더 정보를 스키마 저장소에서 찾는다.
- CIMOM은 프로그램이 이해하는 형태로 변환하여 다른 저장소에 프로바이더 관리 정보를 생성한다.

나. 프로바이더 구동 - ②

- 클라이언트로부터 CIM-XML 오퍼레이션으로 인코딩된 요청이 HTTP 서버로 수신된다.
- CIM 서버는 CIM-XML 오퍼레이션을 디코딩한다.
- CIMOM은 해석된 오퍼레이션에 맞는 프로바이더를 찾는다.
- CIMOM은 해당 프로바이더를 구동한다.

다. 정보 추출 - ②,③

- 프로바이더는 주기적으로 정보를 추출하여 저장소에 저장한다.

- CIMOM은 정보를 저장소에서 추출하여 CIM-XML 오퍼레이션으로 인코딩한다.
- CIMOM은 HTTP 프로토콜에 정보를 실어서 전송한다.

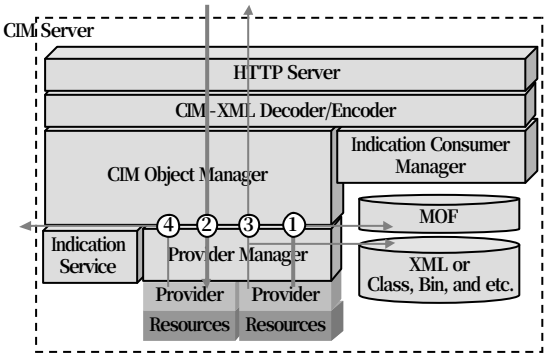
라. 인디케이션 구독신청(Subscription) - ③

- 클라이언트로부터 CIM-XML 오퍼레이션으로 인코딩된 요청이 HTTP 서버로 수신된다. 이때, 요청은 인디케이션 구독 메시지이다.
- CIM 서버는 CIM-XML 오퍼레이션을 디코딩한다.
- CIMOM은 해석된 오퍼레이션에 맞는 프로바이더를 찾아 인디케이션 발행을 요구한다.
- CIMOM은 해당 인디케이션 구독 정보를 저장소에 생성한다.

마. 인디케이션 전송 - ④

- 프로바이더는 인디케이션을 발생시킨다. 발생된 인디케이션을 CIM-XML 오퍼레이션으로 인코딩한다.
- 인코딩된 메시지를 인디케이션 서비스(indication service)를 통해 인디케이션을 수신 받을 클라이언트로 전송한다. 인디케이션 서비스 컴포넌트는 CIMOM의 일부분으로 구현될 수 있으나 CIM 서버의 기능을 보기 쉽게 표현하기 위해 별도의 컴포넌트로 도식화하였다.

(그림 10)은 대표적인 시나리오에 따른 WBEM 구조의 구성요소간 동작을 설명한 것이며 서로의 흐름이 중첩되어 동작할 수 있다.



(그림 10) CIM 서버

Ⅲ. 결 론

본 논문에서는 분산 네트워크 환경에서 시스템 자원을 효과적으로 관리하기 위하여 DMTF가 제시한 표준과 관리 구조 및 요소기술에 대해 기술하였다.

기존의 관리 솔루션들은 플랫폼마다 정보를 추출하는 에이전트를 별도로 개발해야 하지만 WBEM 구조에서는 프로바이더를 통해 별도의 에이전트 구현 없이 최소한의 수정만으로도 이기종의 플랫폼에서도 정보 추출이 가능하다.

현재 CIM/WBEM 기반으로 이를 구현한 오픈소스 프로젝트는 Sourceforge의 WBEMservices, 오픈 그룹의 OpenPegasus, SNIA의 SNIA CIMOM, Center 7의 OpenWBEM 등이 있다. 이 중 OpenPegasus 프로젝트의 활동이 가장 활성화되어 있으며 소스와 개발 문서가 잘 갖추어져 있다.

앞으로 구현될 WBEM은 다음과 같은 요소를 갖추어야 할 것으로 여겨진다[8].

WBEM은 플랫폼 중립적이어야 한다. 이는 어느 특정 플랫폼에 의존적이지 않으며 모든 플랫폼에 대하여 독립적이어야 함을 의미한다. 그리고 WBEM은 언어 중립적이어야 한다. 그러나 현실적으로는 C++이나 자바 언어 등 잘 알려진 객체 언어들로 구현되고 있다. 따라서 언어 중립적이라는 말은 어느 하나의 언어에 종속적이어서는 안된다는 의미가 더 강하다.

WBEM은 표준 데이터 모델에 의존해야 한다. 이는 앞에서도 설명한 정보모델인 CIM을 WBEM이 선택함으로써, 이미 자연스럽게 해결되었다. WBEM은 새로운 표준을 만들기는 하지만 기존의 표준들을 사용하여 만들어 나가야 한다. 이 의미는 기존의 여러 정보 관리 모델들을 통합해야 함을 의미한다.

WBEM은 경량화 및 모듈화 되어야 한다. 앞에서 설명한 바와 같이 WBEM은 성능보다는 여러 표준들을 통합해야 한다는 점에 더 큰 비중을 두는 만큼, 성능보다는 경량화 및 모듈화에 더 큰 비중을 두어야 한다.

WBEM은 데이터 모델과 관련된 모든 시스템 자원들을 포괄적으로 제어할 수 있어야 하며 아주 작은 장치에서부터 규모가 매우 큰 시스템에 이루기까지 모든 분산 환경과 관련된 자원들을 관리할 수 있는 구조적 확장성이 보장되어야 한다.

또한 WBEM은 DMTF에서 정의한 CIM operations를 효율적으로 지원해야 하며, 또한 WBEM의 구성요소들을 시각적으로 보여줄 수 있는 GUI 개발 환경이 구축되어야 할 것이다.

참 고 문 헌

[1] 이금모, "WBEM을 이용한 망 설정 관리 프레임워크," 연세대학교 대학원 전기전자공학과, June 2001.

[2] DMTF, "Common Information Model(CIM) Specification, Version 2.2," June 1999, http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf

[3] Denise Eckstein, "WBEM Overview," www.openpegasus.org/doc.tpl?CALLER=documents.tpl&gdid=3448, July 2003.

[4] Architectural White Paper of DMTF SMWG, www.dmtf.org/apps/org/workgroup/smwgarch/download.php/9211/SMWG_Arch_WP_v1[1].0a_Build0.doc

[5] DMTF, "Specification for CIM Operations over HTTP Ver. 1.0," Aug. 1999, http://www.openpegasus.org/uploads/40/2911/PegasusWorkShop-Architecture.ppt

[6] OpenGroup, "Pegasus Technical Workshop -Pegasus Architecture," Jan. 2000, http://www.dmtf.org/standards/documents/WBEM/DSP200.html

[7] SBLIM, http://www-124.ibm.com/sblim/

[8] 조희남, 안창원, "Analysis of Open Source Project WBEM Implementations," KNOM Review, Vol.7, No.1, Aug. 2004.