# NETCONF 표준을 따른 XML 기반의 네트워크 구성관리 시스템

# (XML-based Network Configuration Management System using NETCONF)

최현미*, 최미정*, 홍원기*, 박용석**

* 포항공과대학교 컴퓨터공학과 분산처리 및 네트워크관리 연구실

** 삼성전자 통신연구센터 IP 연구실

{siwa, mjchoi, jwkhong}@postech.ac.kr, yongseok.park@samsung.com

## 요 약

인터넷이 발전함에 따라 네트워크의 규모는 나날이 증가하고 있으며, 다양한 네트워크 장비들이 출현하고 있다. 이러한 네트워크 장비들은 기본적으로 네트워크 관리를 위하여 SNMP 에이전트를 내장하고 있다. 현재까지 SNMP는 인터넷 관리에서 널리 사용되고 있지만 제한적인 관리 정보 모델링 기법과 제한적인 관리기능 들이 단점으로 지적되고 있다. 특히 그 중에서도 구성관리의 취약점이 많이 지적되고 있다. 이에 대한 해결 방안으로 XML 기반의 네트워크 관리가 대두되고 있다. XML 기반의 네트워크 관리는 현재 구성관리를 중심으로 활발히 연구되고 있다. IETF의 워킹그룹인 NETCONF는 네트워크 장비 업체인 Cisco와 Juniper Networks를 중심으로 XML 기반의 구성관리를 위하여 표준화 작업을 진행하고 있다. 본 논문에서 우리는 NETCONF에서 진행 중인 Internet Draft를 분석하여 문제점을 지적하고, 그에 대한 해결 방안을 제시하고자 한다. 그리고 이러한 문제점을 보완한 XML 기반의 구성관리 시스템(XCMS)을 제안한다. 우리가 구현한 XCMS(Xml-based Configuration Management System)는 전송 프로토콜에 의존적이지 않은 SOAP RPC 메시지를 이용하여 매니저와 에이전트 간의 통신이 이루어지는 구성관리 시스템이다. 본 논문에서는 NETCONF에서 제시한 관리 프로토콜에 기반을 둔 XCMS의 설계 및 구현 방법에 대해 제시하고, 더 나아가 XML 기반의 네트워크 관리 시스템으로 확장할 수 있는 SOAP을 이용한 XML 기반의 구성관리 시스템의 전체적인 구조를 제시한다.

**Keywords**: XML, XML 기반 구성관리, NETCONF, SOAP

## 1. 서론

현재 인터넷의 빠른 확산으로 다양한 네트워크 장치들이 여러 업체들에 의해서 개발되고 있으며, 이러한 장치들로 구성된 거대한 네트워크가 형성되었다. 현재 네트워크 관리를 위해 사용되고 있는 SNMP(Simple Network Management Protocol)[1] 기반의 관리시스템은 나날이 복잡해지고 거대해진 네트워크를 수용하기에는 관리 정보 모델과 관리기능 (특히 구성관리) 측면에서 취약함이 지적되고 있다. 또한 한정된 크기의 UDP를 기반으로 SNMP 메시지를 보내기 때문에 벌크 데이터(bulk data)를 처리 하는데 있어서 가장 큰 문제가 된다[2]. 그래서 이러한 문제를 극복하고 통합적으로 네트워크를 관리할 수 있는 표준화된 네트워크 관리 프로토콜이 요구되고 있다. 그 요구를 충족할 수 있는 SNMP의 대체 방법으로 XML(Extensible Markup Language)[3] 기반의 네트워크 관리가 대두되고 있다. XML 기반의 네트워크 관리는 구성관리(Configuration Management)를 중심으로 IETF의 워킹그룹(WG)인 NETCONF(Network Configuration)[4]에서 표준화 작업을 활발히 진행하고 있다. NETCONF는 관리 오퍼레이션들과 매니저와 에이전트간에 전송되는 메시지 포맷인 관리 프로토콜에 대하여 표준화 하고 있다. 이러한 관리 프로토콜의 표준화 작업은 여러 업체에서 공급되는 다양한 네트워크 장비들의 에이전트들과 매니저간의 관리 오퍼레이션을 위한 상호 운영성을 보장해 주게 된다.

XML은 임의의 엘리먼트(element)와 어트리뷰트(attribute)를 정의할 수 있기 때문에 복잡한 관계를 갖고 있는 정보를 구조적으로 쉽게 표현할 수 있다. 뿐만 아니라 함축적으로 정보의 내용을 내포(self-description) 할 수 있기 때문에 매우 강력한 모델링 언어이다. 이러한 특징을 갖고 있는 XML은 복잡한 관리 정보를 구조적으로 표현할 수 있을 뿐만 아니라 두 호스트간에 전송되는 메시지를 하나의 관리 프로토콜로 정의할 수 있게 한다. 물리적으로 일정하게 정해진 크기의 메시지 포맷을 바이너리(0 또는 1)로 표현하고 있는 기존의 프로토콜 정의 방법은 프로토콜 변경이나 확장성에 제약이 있다. 그러나 프로토콜을 XML 형식의 구조로 정의하면

크기가 고정된 바이너리 표현이 아닌 텍스트로 표현하므로 프로토콜의 변경이나 확장 측면에서 기존의 방법보다 용이하다. 특히 네트워크의 전송 프로토콜(transport protocol)이 아닌 두 호스트간에 전송되는 메시지를 정의하는데 있어서 XML 형식의 표현은 가장 적합한 프로토콜 정의 방법이 된다. 이렇듯 XML은 네트워크 관리 있어서 크게 두 가지 부분에서 사용된다. 하나는 복잡한 관리 정보를 모델링 하는 것이고, 또 다른 하나는 매니저와 에이전트간에 통신하는 메시지 포맷인 관리 프로토콜을 정의하는 것이다.

NETCONF의 관리 프로토콜 메시지는 요청한 서비스를 위해 수행해야 할 오퍼레이션을 바로 호출 할 수 있는 RPC(Remote Procedure Call) 방법을 XML의 태그를 사용하여 정의한다. 즉 정의된 오퍼레이션들의 이름을 XML 태그로 표현하면 그 오퍼레이션 이름을 파서를 통해 추출하고 그 오퍼레이션을 호출해서 수행한다. 그리고 다시 XML 형식의 메시지에 그 결과를 담아서 보내게 된다. 이러한 과정을 통해 구조화된 XML 메시지는 RPC 방법으로 처리가 가능해 진다. 이렇게 RPC를 제공하고 있는 XML 형식의 구조화된 메시지를 W3C(World Wide Web Consortium)에서 표준으로 정한 SOAP(Simple Object Access Protocol)[5] 메시지라고 한다. 이러한 관리 프로토콜의 메시지는 전송 프로토콜의 데이터 (payload) 부분에 들어가기 때문에 전송 프로토콜에 대해 의존적일 필요가 없다. 그러나 전송 프로토콜로 HTTP[6]를 이용하면 데이터 전달의 신뢰성과 많은 데이터를 한번에 보낼 수 있는 장점을 가질 수 있고, 플랫폼에 관계없이 이기종의 시스템 간에도 쉽게 데이터를 교환할 수 있는 장점이 생긴다.

우리는 기존의 XML 관련 툴들을 충분히 활용하여 좀더 쉽고 편리한 구현 환경을 위하여 매니저와 에이전트의 통신 수단으로 SOAP/HTTP를 이용하였다. 본 논문에서는 NETCONF에서 제시한 관리 프로토콜의 문제점 및 해결 방안을 제시하고 이를 기반으로 XPath[7]를 적용한 XCMS의 설계 및 구현 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 진행 중인 NETCONF의 관리 프로토콜[8]에 대한 전반적인 개요와 전송 프로토콜 별로 장단점을 언급한다. 그리고 NETCONF의 관리 프로토콜의 문제점 및 해결 방안을 제시한다. 특히 우리의 XCMS는 SOAP/HTTP를 이용하기 때문에 NETCONF/SOAP I-D[9]의 문제점과 해결책을 제시한다. 3장에서는 XML 기반의 구성관리 시스템을 위한 요구 사항들을 정리하고, 4장에서는 3장의 요구 사항을 바탕으로 XCMS의 전체 구조를 제시한다. 이때 구성관리뿐 아니라 전반적인 네트워크 관리를 위한 시스템으로 확장 가능할 수 있는 통합적인 구조를 제시한다. 5장에서는 4장의 설계 결과에 초점을 맞추어 실제로 우리가 구현한 XCMS에 대해서 설명한다. 마지막으로 6장에서는 결론과 앞으로 가능한 향후 연구 방안에 대해서 언급한다.

## 2. NETCONF 관련 연구

이 장에서는 현재 진행되고 있는 IETF의 워킹그룹인 NETCONF의 표준화 작업에 대해서 소개한다. 그리고 NETCONF의 관리 프로토콜 메시지를 전송하기 위한 전송 프로토콜인 SSH, BEEP, SOAP/HTTP에 대해 별도로 작성된 I-D에 대해 간략히 요약한다. 현재 제안되고 있는 NETCONF 관리 프로토콜의 문제점을 분석하고 그 해결 방안을 제시한다.

### 2.1 NETCONF 프로토콜

NETCONF 프로토콜은 매니저가 네트워크 장비들의 구성 정보를 쉽게 관리할 수 있도록, 그리고 에이전트간의 상호 운용성을 보장하기 위하여 에이전트와 매니저 사이에 전송되는 메시지를 구조화된 XML 형식으로 정의하고 있다. 이러한 구조화된 XML 메시지를 통하여 요청한 관리 서비스는 RPC(Remote Procedure Call) 방법으로 제공된다. 즉, 요청 메시지에 수행하고 싶은 오퍼레이션 이름을 XML 태그 안에 명시하여 보내면 이 메시지를 받은 쪽에서는 자신의 XML 파서를 통하여 오퍼레이션 이름을 추출하고 그 오퍼레이션을 실행시킨 후 그 수행 결과를 응답 메시지에 담아서 보내게 된다.

구성 관리를 수행하고자 하는 네트워크 장비의 전송, 관리 데이터, 관리 오퍼레이션 등의 환경은 매우 다양하다. 그러므로 하나의 전송, 데이터, 오퍼레이션 모델의 정의만으로는 예상되는 다양한 환경의 네트워크 장비에 대한 관리 오퍼레이션의 요구를 명시하기가 쉽지 않다. 따라서 전송되는 메시지를 4가지 계층으로 나누어 정의함으로써 다양한 환경의 관리 요구 사항을 충족시킬 수 있게 된다. 표 1은 NETCONF에서 정의하고 있는 4가지 계층과 그 계층에 포함되는 내용들을 예로 보여주고 있다. NETCONF 프로토콜은 전송되는 메시지 포맷뿐 아니라 'Operation' 계층과 'RPC' 계층에 포함되는 RPC 오퍼레이션들에 대해서도 정의하고 있다.

| 계층 | 내용 |
|---|---|
| Content | XML 형태의 관리 구성 정보를 나타냄<br>Configuration data |
| Operation | NETCONF 에서 정의한 RPC 를 제공할 오퍼레이션들을 나타냄<br><get-config>,<edit-config>, etc. |
| RPC | RPC 오퍼레이션에 대한 요청과 응답을 나타냄 |

| | |
|---|---|
| | <rpc>, <rpc-reply>, etc. |
| Transport | 메시지들을 전달하는 전송 프로토콜<br>BEEP, SSH, HTTP etc. |

**표 1. NETCONF 관리 프로토콜의 계층적 구조**

현재 관리 구성 정보인 Content는 관리되어야 할 장비마다 그 내용이 다르게 정의되기 때문에, 구성 정보 내용은 장비 업체에 의존적이게 된다. 따라서 구성 정보를 정의하는 언어와 그 내용에 대한 표준화 작업은 별도로 이루어져야 한다. NETCONF는 Content 계층 아래 부분에 대한 표준화 작업을 활발히 진행하고 있다. NETCONF에서는 관리 구성 정보의 표준화 작업을 수행하고 있지 않지만, 구성 정보의 상태를 여러 단계로 나누어 구성 정보의 수정에 따른 네트워크 장비의 잘못된 동작을 미연에 방지하고자 한다. 즉, 현재 장비가 수행되고 있는 구성 정보를 'running'이라고 명시하고, 'running' 상태의 구성 정보 내용의 안정성이 검증되면 'startup'이라는 이름으로 현 구성 정보를 저장하여 백업용으로 사용한다. 만약 구성 정보를 수정하고 싶으면 'candidate' 상태에서 수정 작업을 거친 후, 테스트를 통하여 수정된 구성 정보의 무결성이 검증되면 'running' 상태로 진행한다.

NETCONF는 SNMP에서 제공하고 있는 'get', 'set', 'trap'에 대응할 수 있는 오퍼레이션 뿐만 아니라, 트랜잭션(transaction) 처리 즉, 관리 오퍼레이션 수행 중에 에러가 나면 수행 전의 상태로 돌아가는 'rollback' 또는 에러가 난 곳에서 멈추고 에러 메시지를 보내는 'stop-on-error' 등이 정의된다. 또한 구성 정보의 일관성을 유지하기 위해서 여러 매니저가 동시에 한 장비의 구성 정보를 수정하고자 할 때, 수정되고 있는 구성 정보에 대해서 락(lock)을 걸 수 있는 오퍼레이션이 정의된다. 이는 SNMP에서 제공하지 않는 관리 오퍼레이션들이 새롭게 정의 되었음을 알 수 있다. 그리고 NETCONF에서는 매니저가 요청한 메시지에 대해 매니저가 제어할 수 있는 방법을 'RPC' 계층의 태그 정의를 통해서 제공한다. <rpc> 태그 안에 요청 메시지를 명시할 수 있는 'message-id'를 정의하여 이전에 요청한 메시지를 제어할 때 사용한다. 예를 들면 매니저가 특정 'message-id'의 오퍼레이션을 취소(<rpc-abort>)하거나 그 오퍼레이션의 처리 응답 시간이 너무 오래 걸리면 에이전트로부터 그 서비스의 진행 상황을 매니저가 보고(<rpc-progress>) 받을 수 있다.

NETCONF는 계층으로 나누어 정의하여 전송 프로토콜에 대해서는 중립적이다. 따라서 어떠한 전송 프로토콜도 지원하는 것이 기본 개념이다. 'Transport' 계층은 매니저와 에이전트의 연결(connection)에 대해 정의한다. 매니저와 에이전트가 하나의 세션(session)을 형성하면 NETCONF에서는 다중의 TCP 연결이 가능하다. NETCONF에서 이런 TCP 연결을 'channel'이라고 부르고 있으며, 세 가지 종류의 채널(Management, Operation, Notification channels)을 정의하고 있다. Management channel은 'RPC' 계층에 있는 <rpc-abort>, <rpc-progress>와 같은 오퍼레이션이 'Operation' 계층에 있는 관리 오퍼레이션을 제어하기 위해 사용된다. 또한 에이전트가 제공 가능한 관리 기능(capabilities)에 대한 정보를 매니저에게 알릴 때도 사용된다. Operation channel은 주로 'Operation' 계층의 관리 오퍼레이션을 수행할 때 사용하며, Notification channel은 SNMP의 트랩과 같이 에이전트에서 매니저에게 특정 통지(notification)를 보내는 보고 형식의 오퍼레이션을 수행할 때 사용된다. 관리 기능 수행 채널은 반드시 필요한 채널이고 나머지 두 채널의 구현은 선택적이다[8].

## 2.2 NETCONF의 전송 프로토콜

NETCONF의 관리 프로토콜 메시지는 어떠한 전송 프로토콜에서도 전송되어야 한다. 그러나 현재 NETCONF에서는 전송 프로토콜로 SSH, BEEP, SOAP/HTTP에 대해서만 고려한다. NETCONF는 각각의 전송 프로토콜에 대해 별도의 I-D를 제안하고 있다. 이 장에서는 각 I-D를 비교, 분석하여 각 전송 프로토콜의 장단점을 요약 정리하고자 한다.

### 2.2.1 SSH

SSH(Secure Shell)[10]는 기존의 네트워크 장비의 관리에 있어서 주로 사용해 오던 환경이다. 따라서 기본적으로 네트워크 오퍼레이터들의 강력한 요구에 의해 제공되어야 하는 전송 프로토콜이다. 그러나 SSH의 I-D는 구현의 간편성을 위하여 관리 수행을 위한 하나의 TCP연결만을 제안하고 있다. SSH의 경우 에이전트가 매니저에게 메시지를 비동기적으로 보낼 수 있는 방법이 제공되지 않기 때문에 보고 채널 같은 경우는 유지할 수도 없다[11].

### 2.2.2 BEEP

BEEP(Block Extensible Exchange Protocol)[12]은 현재 Cisco와 Juniper Networks와 같은 대표적인 네트워크 장비 업체에서 추천하고 있는 전송 프로토콜이다. BEEP I-D는 SSH I-D와 달리 NETCONF에서 정의한 다중 채널을 지원하도록 제안하고 있다. BEEP은 서버/클라이언트의 구조가 아닌 peer의 구조를 가지기 때문에 매니저가 에이전트에게 또는 에이전트가 매니저에게 관리 정보를 요청할 수 있다. 따라서 에이전트로부터 매니저에게 전송되는 비동기적 메시지를 쉽게 처리할 수 있는 장점이 있다[13].

### 2.2.3 SOAP over HTTP

SOAP은 NETCONF의 의도에 가장 적합한 프로토콜이라고 할 수 있다. 왜냐하면 SOAP은 자체적으로 RPC 메카니즘을 제공하고 있으며, SOAP RPC 메시지는 다양한 전송 프로토콜을 지원할 수 있기 때문이다. 게다가 SOAP을 지원하는 관련 툴들이 많이 제공되므로 시스템 구현 측면에서 다른 프로토콜보다 매우 용이하다. 그러나 NETCONF에서 전송 프로토콜을 SOAP 하나로 제한하지 못하는 것은 현재 SOAP을 제공하고 있는 구현 환경이 전송 프로토콜로 주로 사용하는 HTTP는 서버/클라이언트 구조를 따르고 있어서 서버 역할을 하고 있는 에이전트가 매니저에게 메시지를 전송하는데 문제가 생기기 때문이다. 이러한 문제는 에이전트가 매니저에게 비동기적으로 보내는 메시지 처리에 제약을 준다. 이를 해결하기 위한 방법으로 매니저에서 주기적으로 에이전트에게 폴링하여 통보 메시지를 가져오는 방법이 있지만 매번 주기적으로 폴링을 해야 하기 때문에 불필요한 트래픽과 매니저의 수행 오버헤드를 발생 시킨다. 또 다른 해결책으로 현재까지 가능성은 없지만 HTTP대신 BEEP을 이용한 SOAP 구현 환경이 개발되는 것이다[9].

## 2.3 NETCONF의 문제점 및 해결 방안

현재 NETCONF 프로토콜에서는 관리 오퍼레이션을 수행하기 위해 선택하려는 구성 정보를 나타내는데 어려움이 있다. NETCONF 프로토콜의 기존 방법은 선택하고 싶은 노드의 위치를 정확하게 표시할 수 없기 때문에 선택할 구성 정보의 태그를 나타내기 위하여 그 태그 이전의 모든 상위 부모 노드의 태그까지 서술한다. <get-config> 오퍼레이션의 경우, 선택하고 싶은 구성 정보를 정확히 표현 할 수 없기 때문에 오퍼레이션 수행 결과가 선택되지 않은 구성 정보와 함께 보여주기도 한다. <edit-config>의 경우는 변경하고 싶은 구성 정보를 정확하게 표현해야 하기 때문에 수정하고 싶은 구성 정보 사이에 오퍼레이션의 어트리뷰트(merge, replace, delete)의 값을 삽입시켜야 한다. 이는 수행해야 할 오퍼레이션을 호출하기 위해서 수정하고 싶은 구성 정보 사이에 있는 오퍼레이션 어트리뷰트 값을 추출해야 하는 과정이 요구된다.

이 문제점을 해결하기 위해서 우리는 XML 데이터의 노드의 위치를 명료하게 표현할 수 있는 표준 방법인 XPath[7]의 사용을 제안한다. XPath를 사용하게 되면 <get-config>의 경우는 선택하고 싶은 구성 정보만을 가져올 수 있고, <edit-config>의 경우는 수정하고 싶은 구성 정보들 사이에 오퍼레이션 어트리뷰트 값을 삽입할 필요없이 <edit-config-{operation}> 표현을 통해 오퍼레이션을 호출할 수 있게 된다. 이때 {operation} 안에 들어갈 값은 ‘merge’, ‘replace’ 그리고 ‘delete’가 된다. 표 2는 기존 NETCONF 프로토콜 I-D에서 제시한 <edit-config>의 예와 우리가 제시한 XPath를 사용한 표현을 비교하여 나타낸 것이다.

**XPath를 사용하지 않은 <edit-config>의 replace**

```
<rpc message-id="107"
  mlns="http://ietf.org/xmlconf/1.0/base">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns="http://example.com/schema/1.2/config"
            xmlns:xc="http://ietf.org/xmlconf/1.0/base">
      <interface>
        <name>Ethernet0/0</name>
        <mtu>1500</mtu>
        <address xc:operation="replace">
          <name>1.2.3.4</name>
          <mask>255.0.0.0</mask>
        </address>
      </interface>
    </config>
  </edit-config>
</rpc>
```

**XPath를 사용한 <edit-config-replace>**

```
<rpc message-id="107"
  xmlns="http://ietf.org/xmlconf/1.0/base">
  <edit-config-replace>
    <target>
      <running/>
    </target>
    <xpath>
      //interface/name[.="Ehernet0/0"]/following::mtu[.="1500"]
      /following::address
    </xpath>
    <config xmlns="http://example.com/schema/1.2/config"
            xmlns:xc="http://ietf.org/xmlconf/1.0/base">
      <address>
        <name>1.2.3.4</name>
        <mask>255.0.0.0</mask>
      </address>
    </config>
  </edit-config-replace>
</rpc>
```

**표 2. XPath를 사용한 <edfi-config>의 예**

NETCONF 관리 프로토콜은 XML의 구조적인 메시지를 통하여 NETCONF에서 정의한 오퍼레이션들을 호출하여 구성 관리를 하고 있다. 이는 RPC를 제공하고 있지 않은 메커니즘에서도 XML 형식의 구조화된 메시지의 파싱을 통하여 RPC 효과를 충분히 얻을 수 있게 된다. 그러나 SOAP처럼 RPC 메카니즘을 제공하고 있는 환경에서는 NETCONF 관리 프로토콜의 메시지의 분석을 통해 실제로 호출하고자 하는 오퍼레이션이 무엇인지, 그리고 그 오퍼레이션의 파라미터들이 무엇인지에 대한 정의가 명확히 이루어져야 한다. 이러한 오퍼레이션에 필요한 정보들은 표준 WSDL(Web Services Description Language)[12]을 통해 명세 할 수 있다.

**<rpc> 태그를 포함하는 <get-config>의 SOAP 메시지**

```
<?xml version="1.0" encoding="UTF-8"?>
    <soapenv:Envelope

xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
      <soapenv:Body>
       <rpc id="101" xmlns="http://ietf.org/netconf/1.0/base">
        <get-config>
         <source>
          <running/>
         </source>
         <config
xmlns="http://example.com/schema/1.2/config">
          <users/>
         </config>
         <format>xml</format>
        </get-config>
       </rpc>
      </soapenv:Body>
    </soapenv:Envelope>
```

**<rpc> 태그를 생략한 <get-config>의 SOAP 메시지**

```
<?xml version="1.0" encoding="UTF-8"?>
    <SOAP-ENV:Envelope

xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Body id="_0">   // -? replace <rpc>
element
        <get-config>
          <source><running/></source>
          <xpath>
                //users
          </xpath>
          <format>xml</format>
        </get-config>
     </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

**표 3. SOAP 메시지의 예**

WSDL은 RPC 오퍼레이션의 파라미터들의 이름과 데이터 타입, 그리고 오퍼레이션의 수행 결과 값의 데이터 타입에 대한 정보를 기술한다. NETCONF/SOAP I-D에서 제시한 WSDL을 보면, NETCONF 프로토콜의 메시지를 SOAP의 바디(BODY) 부분에 넣기 때문에 NETCONF의 'RPC' 계층에서 정의한 오퍼레이션만이 RPC 오퍼레이션으로 제공된다. 구성 관리를 위해 실제로 자주 쓰이는 'Operation' 계층에 있는 <get-config>, <edit-config> 같은 오퍼레이션들은 'RPC' 계층의 <rpc> 태그에 포함되기 때문에 RPC 오퍼레이션으로 제공되지 못하고 파라미터로 넘겨진다. 이것은 파라미터로 넘겨진 오퍼레이션의 이름을 추출하기 위해 여전히 XML 문서의 파싱 작업이 필요하게 된다. 이는 다른 전송 프로토콜과 비교 했을 때 RPC를 제공하는 SOAP의 장점을 제대로 이용하지 못하는 것이다.

우리는 NETCONF/SOAP에서 효과적으로 SOAP RPC 메시지를 사용하기 위해 'Operation' 계층의 오퍼레이션을 감싸고 있는 <rpc> 태그를 생략해서 SOAP 메시지를 제안한다. 그리고 모든 RPC 오퍼레이션들에 대한 WSDL을 정의한다. 우리가 정의한 WSDL은 NETCONF/SOAP의 관리 오퍼레이션을 정의하기 위한 가이드라인이 될 수 있다. 그리고 SOAP/HTTP에서 비동기적 메시지 처리에 대한 어려움을 극복하기 위해 에이전트와 매니저에 HTTP 서버/클라이언트 모듈을 내장시킨다. 에이전트 내에 있는 HTTP 클라이언트를 통해 에이전트는 매니저에게 비동기적 메시지를 보낼 수 있게 된다. 표 3은 NETCONF의 <get-config> 관리 오퍼레이션 메시지를 현재 NETCONF/SOAP I-D에서 제안하고 있는 <rpc> 태그를 포함하여 SOAP 메시지로 표현한 예와 RPC 오퍼레이션을 고려하여 우리가 제안한 <rpc> 태그를 생략한 SOAP 메시지의 예를 보여주고 있다.

NETCONF/SOAP I-D에서는 매니저가 에이전트에게 요청 메시지를 보낼 때 세션 아이디(session-id)의 값이 비워져 있으면 새로운 세션을 형성하도록 제안하고 있다. 에이전트는 매니저로부터 오는 메시지를 받을 때 마다 매번 세션 아이디 값을 체크해야 하는 파싱 작업을 수행해야 한다. 이런 추가적인 작업을 없애기 위해서 우리는 <create-session> 오퍼레이션을 추가적으로 제안하고, 이 오퍼레이션을 통하여 새로운 세션을 형성하고자 한다. 기존 NETCONF 프로토콜 I-D에서는 세션 아이디 처리를 위한 구체적인 방법에 대해서 언급하고 있지 않다. 우리는 세션 아이디의 정보를 일률적으로 처리할 수 있도록 세션 아이디의 정보를 NETCONF 관리 프로토콜 메시지에 명시하도록 제안한다.

## 3. 요구 사항

이 장에서는 NETCONF의 요구 사항과 이를 바탕으로 우리의 XML 기반의 구성관리 시스템인 XCMS를 위해 더 추가해야 할 요구 사항에 대해 살펴본다.

### 3.1 NETCONF의 요구 사항

NETCONF의 목적은 다양한 네트워크 장비의 구성정보를 효과적으로 관리하기 위해 쉽고 간단한 방법을 제공하는 것이다. 이를 위해 현재 NETCONF에서는 다음과 같은 요구 사항을 제시하고 있다.

(1) 매니저와 에이전트간의 인터랙션(interation)을 위하여 XML 형식의 메시지 구조를 정의한다. 이러한 NETCONF 관리 프로토콜 메시지를 XML로 정의하기 때문에 프로토콜의 확장성과 유연성이 용이하게 된다.

(2) 여러 업체들에 의해서 출시되는 다양한 네트워크 장비에 내장되어 있는 에이전트들에 대해 상호 운영성을 보장해야 한다.

(3) 특정 전송 프로토콜에 제한을 두지 않는다. 이것은 전송 프로토콜에 의해서 전해지는 데이터 부분에 NETCONF 관리 프로토콜

메시지를 전달하기 때문에 가능해 진다.

(4) 매니저와 에이전트의 쉬운 개발 환경을 제공하기 위해 기존의 XML관련 툴들이나 구현된 소스들을 활용해야 한다.

(5) 수정되고 있는 구성 정보에 락(lock)을 걸 수 있게 한다. 이는 여러 매니저가 동시에 같은 구성 정보를 수정 하려고 할 때 발생할 수 있는 문제점을 미연에 방지해 준다.

(6) 명령어를 실행 시킬 수 있는 CLI(Command Line Interface)를 제공한다.

(7) 하나의 매니저에서 여러 대의 에이전트를 관리할 수 있어야 한다.

### 3.2 XCMS의 요구 사항

XCMS는 기본적으로 3.1장에서 언급한 NETCONF의 요구 사항을 만족해야 한다. SOAP/HTTP의 통신을 이용하고 있는 XCMS의 추가적인 고려 사항들은 아래와 같다.

(1) 매니저가 에이전트에게 구성 정보를 업 로딩 할 수 있거나 다운 로딩되어야 한다. 이는 매니저가 관리하고 있는 모든 에이전트의 구성 정보를 중앙에서 일괄적으로 처리 할 수 있게 된다.

(2) 하나의 세션에 다중 채널(Management, Operation, Notification channels)을 기본적으로 제공한다. 그리고 에이전트는 알림 메시지 또는 <rpc-progress>같은 비동기적 메시지를 매니저에게 전송 할 수 있어야 한다. .

(3) 구성 정보의 상태를 여러 단계로 나누어 구성 정보의 수정에 따른 네트워크 장비의 잘못된 동작을 미연에 방지 한다.

(4) XCMS는 SOAP/HTTP 구현 환경을 이용하기 때문에, 각 RPC 오퍼레이션에 대한 WSDL를 정의한다.

(5) 세션 아이디의 정보를 일률적으로 처리할 수 있도록 세션 아이디의 정보를 관리 프로토콜 메시지에 명시한다.

(6) XPath 처리를 위한 파서가 지원되어야 한다.

(7) XML기반의 네트워크 관리 시스템으로 확장할 수 있는 SOAP 구현 환경을 이용한 XML기반의 구성 관리 시스템의 전체적인 구조를 제시한다

관리 기능을 수행하는 에이전트는 각 장비가 갖는 고유한 기능에 장애를 주지 않도록 작고 효율적이어야 한다. 그리고 어느 장비에든 탑재되어 관리 기능을 제공할 수 있도록 이식성이 좋아야 한다.

## 4. 설계

이 장에서는 3장의 요구 사항을 바탕으로 XCMS를 설계한다. 먼저 4.1장에서는 관리 프로토콜을 제시하고 4.2장에서는 트랜잭션 처리와 알림 메시지 저장을 위한 로그 정보 모델을 제시한다. 마지막으로 4.3장에서는 XCMS의 전체적인 구조에 대해서 설명한다.

### 4.1 구성 관리 프로토콜 모델

XCMS는 SOAP 메시지를 이용하여 매니저와 에이전트간의 통신이 이루어지고 있다. 효과적으로 SOAP RPC 메시지를 사용하기 위해 NETCONF의 'RPC' 계층과 'Operation' 계층으로 나누지 않고 아래 표 4와 같이 세 계층으로 나타낸다.

| XCMS Level | SOAP Mgs | Example |
|---|---|---|
| Content | Parameters | Configuration data |
| Operation | SOAP RPC Operation | <get-config>,<edit-config-merge>,<lock>,<unlock>,<create-session>,<kill-session>, <notification>,<rpc-abort>, <rpc-progress>, etc |
| Transport | HTTP | HTTP |

**표 4. XCMS 관리 프로토콜의 계층**

SOAP RPC 메시지에서는 RPC 오퍼레이션과 파라미터의 정보가 중요하다. SOAP 메시지 측면에서 NETCONF의 'RPC' 계층과 'Operation' 계층에서 제공되는 오퍼레이션들은 SOAP RPC 오퍼레이션으로 인식된다. 결과적으로 'RPC' 계층에 있는 <rpc> 태그를 생략하게 되는데, <rpc> 태그는 NETCOF의 'RPC' 계층을 표시하기 위한 역할일 뿐 오퍼레이션을 의미하지 않기 때문에 SOAP RPC 메시지에서 <rpc> 태그를 생략하였다.

| 구성 관리 프로토콜의 XML구조 | Example |
|---|---|
| <RPC Operation Name><br><parameter 1><br></parameter1><br><parameter 2><br></parameter2><br><parameter 3><br></parameter3><br>…<br></RPC Operation Name> | <edit-config-replace><br><target><br><running/><br></target><br><xpath> //interface</xpath><br><config><br><interface><br><name>Ethernet</name><br><mtu>1500</mtu><br><address><br><name>1.2.3.4</name><br><mask>255.0.0.0</mask><br></address><br></interface><br></config><br><transaction>rollback</transaction><br><sessionID>234</sessionID><br></edit-config-replace> |

**표 5. XCMS 관리 프로토콜**

표 5는 XCMS의 매니저와 에이전트 사이에 전송되는 요청 메시지의 XML 구조와 예를 보여준다. XCMS의 구성관리 프로토콜의 요청 메시지는 SOAP RPC 메시지의 형식을 따르고

있으며 예제에서 알 수 있듯이 XPath의 사용은 기존의 <edit-config> 오퍼레이션의 구조에 변화를 준다. XPath는 필요한 구성 정보만을 가지고 원하는 노드 위치를 명확하게 표시할 수 있기 때문에 기존의 NETCONF 프로토콜의 I-D에서 제시하는 방법과 같이 특정 구성 정보 내에 오퍼레이션 어트리뷰트(merge, replace, delete) 값을 삽입시키지 않아도 된다. <edit-config-{operation}>으로 새롭게 정의한 오퍼레이션은 구성 정보로부터 오퍼레이션 어트리뷰트 값을 파싱할 필요없이 매니저가 에이전트의 오퍼레이션을 직접 호출할 수 있게 된다. 이 때 {operation}이라고 표시된 값에는 기존의 오퍼레이션 어트리뷰트 값인 'merge', 'replace', 'delete'가 삽입된다.

**<edit-config-replace>의 WSDL**

```
<?xml version="1.0" encoding="UTF-8"?>
 <definitions name="netconf"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://ietf.org/netconf/1.0/soap"
  xmlns:xb="http://ietf.org/netconf/1.0/base"
  targetNamespace="http://ietf.org/netconf/1.0/soap"
  name="http://ietf.org/netconf/1.0/soap">
 <import namespace="http://ietf.org/netconf/1.0/base"
          location="base.xsd"/>
 <message name="edit-config-replaceRequest">
  <part name="target" type="xsd:string"/>
  <part name="xpath" type="xsd:string"/>
   <part name="config" type="xsd:string"/>
   <part name="transaction" type="xsd:string"/>
   <part name="sessionID" type="xsd:string"/>
 </message>
 <message name="edit-config-replaceResponse">
  <part name="rpc-reply" type="xsd:string"/>
 </message>
 <portType name="netconfPortType">
  <operation name="edit-config-replace">
    <input message="tns: edit-config-replaceRequest"/>
    <output message="tns: edit-config-replaceResponse"/>
  </operation>
 </portType>
 <binding name="rpcBinding" type="tns:rpcPortType">
  <SOAP:binding style="rpc"
   transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="edit-config-replace">
   <SOAP:operation/>
    <input>
     <SOAP:body use="encoded"
      namespace="http://ietf.org/netconf/1.0/base"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
     <SOAP:body use="encoded"
      namespace="http://ietf.org/netconf/1.0/base"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
   </operation>
  </binding>
 </definitions>
```

**표 6. <edit-config-replace> 오퍼레이션의 WSDL**

WSDL은 SOAP 구현 환경에서 제공하는 툴을 이용하여 자동으로 생성된다. 표 6은 툴을 이용하여 코드로부터 자동 생성된 <edit-config-replace> 오퍼레이션의 WSDL을 보여준다. SOAP RPC의 사용은 RPC 오퍼레이션의 명세가 필요하고, 각 RPC 오퍼레이션마다 WSDL를 정의해야 한다. 이는 WSDL 정의대로 에이전트의 RPC 오퍼레이션들을 개발하게 되면, SOAP 구현 환경의 에이전트들간에 상호 운영성이 보장 된다.

### 4.2 로그 정보 모델

XCMS는 네트워크 구성 관리를 위해 트랜잭션 처리에 대한 로그 정보와 에이전트로부터 전달 받은 알림(notification) 메시지에 대한 로그 정보를 갖게 된다. 트랜잭션 처리는 <edit-config-{operation}>의 관리 오퍼레이션을 요청할 때 파라미터(표 3의 Example참고)로 넘겨지게 된다. XCMS는 트랜잭션을 처리를 위해 다음과 같은 두 가지 경우를 정의하였다. 'rollback'은 수행 중 에러가 발생하면 예전 값으로 돌아가는 것이고, 'stop-on-error'는 에러가 발생한 곳에서 멈추어 더 이상 수행하지 않는 것이다. 이 경우에는 변경된 값을 그대로 유지하게 된다. 그리고 XCMS에서 알림 메시지는 에이전트를 리부팅할 때와 에이전트 시스템을 정상적으로 종료할 때 발생된다. 표 7은 로그 정보에 대한 예를 보여주고 있다.

**Transaction처리에 대한 로그 정보**

```
<transaction>
 <agent>
  <ip>121.22.22.22</ip>
  <operation>edit-config-replace</operation>
  <time>2003:08:10:12:05:02</time>
  <reply>success< /reply>
 </agent>
  …
</transaction>
```

**Notificaton 로그 정보**

```
<notification>
 <agent>
  <ip>121.22.22.22</ip>
  <mgs>reboot</mgs>
  <time>2003:08:10:12:05:02</time>
 </agent>
 …
</ notification >
```

**표 7. 로그 정보 모델**

### 4.3 XCMS의 구조

NETCONF/SOAP I-D에서 제시한 비동기적 메시지 처리에 대한 어려움을 해결하기 위해 XCMS는 HTTP 서버/클라이언트를 매니저와 에이전트 모두에 내장한다. XCMS는 하나의 세션에 기본적으로 세 개의 채널을 제공한다. Notification channel을 통해 에이전트로부터 오는 알림 메시지와 같은 비동기적 메시지를 매니저가 처리 하기 위해

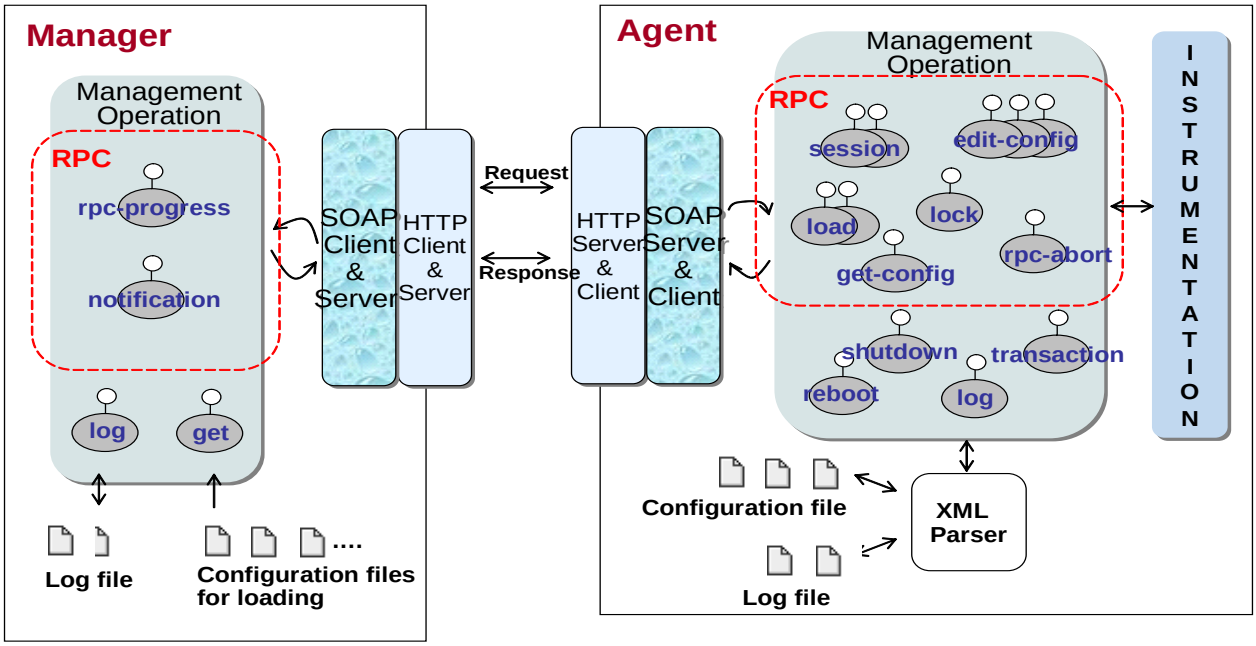



**그림 1. XCMS 구조**

매니저는 HTTP 서버를 포함한다. 그리고 에이전트는 서로 다른 TCP 연결을 위한 두 채널(Management, Operation channels)을 제공하기 위해 HTTP 서버를 포함한다. 이렇게 각 에이전트와 매니저에서 수행되는 HTTP 서버를 통해 원격으로 RPC 오퍼레이션들의 웹 서비스(Web Services)가 SOAP 모듈의 서버와 연동한다. 만약 네트워크 관리를 위해 새로운 관리 오퍼레이션을 추가하고자 한다면 WSDL로 정의된 새로운 RPC 오퍼레이션을 새로운 RPC 오퍼레이션을 추가하고 그에 따른 수행 코드를 작성하면 된다.

그림 1은 XCMS의 매니저와 에이전트의 구조를 보여주고 있다. XCMS 매니저는 CLI에서 사용자의 명령어를 받아 실행되기 때문에 간단한 매니저 구조를 보이고 있다. 매니저는 크게 관리 오퍼레이션 모듈과 SOAP 모듈 그리고 로그 파일(트랜잭션 처리에 대한 로그 정보, 알림 로그 정보)들과 에이전트에게 원격으로 로딩할 구성 정보들의 파일들로 구성된다. 그리고 관리 오퍼레이션은 에이전트에게 원격으로 불려질 RPC 오퍼레이션(<rpc-progress>, <notification>)과 로컬에서 불려질 오퍼레이션(<log>, <get>)들로 나뉘게 된다. 아래는 매니저 관리 오퍼레이션들에 대해 간단히 설명하고 있다.

- <rpc-progress>: 에이전트가 요청 받은 RPC 서비스를 처리하는데 일정 시간이 지나도 서비스 처리가 끝나지 않을 경우 에이전트가 매니저에게 서비스 처리의 진행률을 알려 줄 때 사용된다.
- <notification>: 에이전트 시스템이 정상적으로 종료되거나 에이전트의 새로운 구성 정보를 시스템에 적용 시키기 위해 에이전트를 리부팅 할 때 에이전트에 의해서 호출된다.
- <get>: 매니저가 자신의 CLI에서 매니저의 로그 정보라든가 구성 정보를 사용자에게 보여줄 때 호출 된다.
- <log>: 로그정보들을 파일로 저장한다.

XCMS의 에이전트는 매니저의 모듈 구조에 XML 파서 모듈을 하나 더 추가시킨다. 그리고 매니저와 달리 에이전트가 갖고 있는 세 종류의 구성 정보 파일은 NETCONF에서 정의한 구성 정보의 세 단계 'candidate', 'running', 'startup'을 의미한다. 에이전트의 관리 오퍼레이션 모듈은 NETCONF의 기본 오퍼레이션들과 그 외 우리가 XCMS를 위해 추가한 오퍼레이션들로 구성된다. 매니저의 관리 오퍼레이션처럼 에이전트의 관리 오퍼레이션도 RPC 오퍼레이션들과 로컬 오퍼레이션들로 나뉘게 된다. 에이전트의 RPC 오퍼레이션들 중에서 <rpc-abort>는 management channel에서 불려지고 그 외 나머지 RPC 오퍼레이션들은 operation channel에서 불려진다. 아래는 에이전트 관리 오퍼레이션들에 대해 간단히 설명하고 있다.

- <load>: 매니저가 에이전트에게 구성 정보를 원격으로 로딩 할 때 불려지는 오퍼레이션들을 나타낸다. <download> 오퍼레이션은 에이전트의 구성 정보를 매니저에 저장하고자 할 때 호출되고, <upload> 오퍼레이션은

매니저가 갖고 있는 구성 정보를 에이전트에 저장할 때 이에 호출 된다.

- <session>: 세션과 관련된 오퍼레이션들을 나타낸다. 즉, 매니저와 에이전트가 새로운 세션을 열 때 불려지는 <create-session> 오퍼레이션과 특정 세션을 닫을 때 수행하는 <kill-session> 오퍼레이션을 의미한다. 세션을 닫을 때, 그 세션과 연결된 모든 채널들도 역시 닫히게 된다. 그리고 세션 아이디는 에이전트에서 유일한 값으로 만들어 져서 매니저에게 보내진다.
- <get-config>: 에이전트가 갖고 있는 구성 정보를 매니저에게 전송하고자 할 때 호출 된다.
- <edit-config-{operation}>: 에이전트가 갖고 있는 구성 정보를 변경할 때 수행되는 세 가지 오퍼레이션을 의미한다. <edit-config-merge>는 새로운 구성 정보를 추가할 때, <edit-config-replace>는 기존 구성 정보를 변경 할 때, <edit-config-delete>는 기존 구성 정보를 삭제 할 때 호출 된다. 그리고 새로 구성 정보를 추가 할 때 파라미터로 넘겨지는 XPath는 추가될 노드의 위치를 가리키고, 구성 정보를 변경하거나 삭제할 때 XPath는 변경되거나 삭제될 구성 정보의 루트 노드를 가리킨다.
- <lock>: 세션 아이디를 가지고 특정 구성 정보에 락을 건다.
- <rpc-abort>: 매니저가 요청한 RPC 서비스를 취소 할 때 호출된다. 이는 요청한 RPC 메시지 아이디(message-id)를 이용하여 취소한다.
- <reboot>: 에이전트가 변경된 구성 정보를 시스템에 적용시키기 위해 리부팅 할 때 불려진다.
- <shutdown>: 에이전트의 시스템이 정상적으로 종료되었을 때 불려진다.
- <log>: 로그정보들을 파일로 저장한다
- <transaction>: <edit-config-{operation}>을 수행할 때 파라미터로 넘겨진 transaction의 값을 보고 불려지는 오퍼레이션이다.

## 5. 구현

우리는 4장에서 제시한 구조를 바탕으로 IP 게이트웨이 장비의 구성 정보를 관리하는 XCMS를 구현 하였다. 이번 논문에서는 NETCONF에서 진행되고 있는 관리 프로토콜을 기반으로 프로토타입의 XML 기반의 구성관리 시스템 구현에 초점을 맞추었다.

### 5.1 구현 환경

어떤 임베디드 시스템에도 탑재할 수 있도록 이식성을 향상시키기 위하여 대부분의 임베디드 시스템에서 일반적으로 사용하는 C 언어를 사용하여, 각 모듈별로 컴포넌트화 하였다. 우리가 XCMS 에이전트를 탑재하여 관리 기능을 제공할 IP 게이트웨이 장비는 모토로라의 MPC850DE[13] 프로세서를 사용하는 16MB 메모리를 장착한 인터넷 공유기이다. 운영 체제로는 linux-2.2.13-7커널 기반의 임베디드 리눅스를 이용하고 있으며, powerpc-linux-gcc 컴파일러를 사용하여 XCMS의 에이전트가 개발되었다.

매니저는 에이전트에 비해서 컴퓨팅 자원에 대한 문제가 없으므로, 자바에서 제공되는 다양한 XML관련 API를 이용하는 것이 효율적이므로 자바 플랫폼을 사용하였다. Apache[15] 그룹에서 제공하고 있는 자바 환경의 AXIS[16]가 작고 성능이 좋으므로 매니저에 들어갈 SOAP/HTTP 소스로 AXIS를 선택하였다. 에이전트의 경우는 컴퓨팅 자원을 최대한 아껴야 하므로 C 기반의 XML 파서와 SOAP 엔진을 사용하였다. gSOAP[14]은 C/C++환경에서 제공하고 있는 SOAP/HTTP 소스들 중에서 가장 작고 효율적인 SOAP 구현 환경을 제공한다. 또한 다른 SOAP 구현물과 상호 운영성을 보장하고 있다. 따라서 gSOAP은 매니저의 AXIS와 상호 운영성을 보장하므로 매니저와 에이전트간의 SOAP메시지 전송이 가능하다. gSOAP은 RPC 오퍼레이션을 선언한 헤더 파일을 통하여 자동으로 스텁(Stub)과 스켈리톤(Skeleton), 그리고 WSDL을 생성한다. 또한 매니저와 에이전트 사이에 오가는 SOAP RPC 메시지도 자동으로 인코딩되어 보내진다. 이처럼 SOAP 구현 환경에서 제공되는 툴들을 이용하면 시스템 구현 측면에서 효율적이고 편리해진다. 그리고 에이전트와 매니저에 들어갈 SOAP 서버 모듈은 gSOAP에서 내장되어 있는 Stand-alone 서버를 통하여 RPC 오퍼레이션 서비스인 웹 서비스(Web Services)를 C/C++환경에서 제공 한다.

에이전트에 제공되고 있는 XML 파서는 기존 파서들 중에서 테스트를 거친 가장 작고 효율적인 파서인 LIBXML[17]을 이용하여 구현하였다.

### 5.2 CLI

CLI에서 사용자 명령어를 통하여 실제로 에이전트의 구성정보와 관련된 RPC 오퍼레이션들이 호출되는 과정을 보여준다. 구성정보와 관련된 기본 명령어는 다음과 같다.

형식) operationName [hostIP] [database] [xpath] [config]
예) get_config chicago.postech.ac.kr running //admin

operationName : get_config, edit_config-merge,
edit_config_replace, edit_config_delete
hostIP : 관리할 에이전트의 IP 주소
database: 에이전트가 관리하고 있는 세가지 종류의
데이터베이스(candidate, running, startup)

xpath: 원하는 노드의 위치를 XPath 로 표현
config: <edit-config-{merge, replace}> 수행 시 새롭게 넣어주거나 변경될 구성정보

그림 2는 CLI 화면을 보여주고 있다. 아래의 화면은 <get-config>, <edit-config-delete>, <edit-config-merge>, <edit-config-replace> 오퍼레이션들을 수행한 결과들을 보여주고 있다. <edit-config-{operation}>의 수행 후 변경된 구성정보를 확인하기 위해서 <get-config>를 호출하여 변경된 결과를 보여주고 있다.


**그림 2. CLI에서 실행되는 XCMS**

## 5. 결론

현재 NETCONF는 복잡한 네트워크 구성관리를 위한 관리 프로토콜을 XML을 이용하여 정의하는 것에 대한 표준화 작업을 수행하고 있다. 우리는 본 논문에서 기존 NETCONF 프로토콜 I-D들의 문제점을 분석하고, 그 문제점을 해결하는 방법을 제안하였다. 또한, 그 해결 방안을 따르는 XML 기반의 구성관리를 위한 XCMS의 구조를 제시하였으며, IP 게이트웨이에 직접 적용하여 XCMS 기능적인 효용성을 입증하였다.

향후 연구 과제로는 적은 자원을 사용하는 작고 효율적인 XCMS를 제공하기 위해 XCMS 에이전트를 최적화하는 작업이 필요하다. 또한 성능 평가를 통하여 우리가 최적화 한 XCMS의 효율성을 증명해야 한다.

## 참고 문헌


[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin (Eds.), “A Simple Network Management Protocol(SNMP)”, RFC 1157, IETF, May 1990.

[2] J. Schonwalder, A. Pras, J.P. Martin-Flatin, “On the Future of Internet Management Technologies”, IEEE Communications Magazine, October 2003, pp.90~97.

[3] Tim Bray, Jean Paoli and C. M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0”, W3 Recommendation REC-xml-19980210, February 1998.

[4] IETF, “Network Configuration (Netconf)”, http://www.ietf.org/html.charters/netconf-charter.html.

[5] W3C, “SOAP Version 1.2 Part 2: Adjuncts”, W3C Working Draft, December 2001.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1”, RFC 2616, IETF HTTP WG, June 1999.

[7] W3C, “XML Path Language (XPath) Version 2.0”, W3C Working Draft, April 2002.

[8] Enns, R., "NETCONF Configuration Protocol",draft-ietf-netconf-prot-00 (work in progress), Aug 2003,<http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot- 00.txt>.

[9] Goddard, T., "A SOAP Binding for NETCONF", draft-ietf-netconf-soap-00 (work in progress), Oct 2003, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-soap-00.txt>.

[10] Ylone, T., Kivinen, T., Saarinen, M., Rinne, T. and S. Lehtinen, “SSH Transport Layer Protocol”, draft-ietf-secsh-transport-16 (work in progress), July 2003.

[11] Wasserman, M., “Using the NETCONF Configuration Protocol over Secure Shell(SSH)”, draft-ietf-netconf-ssh-00 (work in progress), Oct 2003, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-ssh-00.txt>.

[12] Rose, M., “The Blocks Extensible Exchange Protocol Core”, RFC 3080, March 2001.

[13] Lear, E., Crozier, K., Enns, R., “NETCONF Transport over BEEP”, draft-lear-netconfbeep-00 (work in progress), Aug 2003, <http://www.ietf.org/internet-drafts/draft-lear-netconfbeep-00.txt>.

[14] W3C, “Web Services Description Language (WSDL) Version 1.2” W3C Working Draft, July 2002.

[15] H. T. Ju, M. J. Choi, S. H. Han, Y. J. Oh, J. H. Yoon, H. J. Lee, and J. W. Hong, “An Embedded Web Server Architecture for XML-based Network Management”, Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April 2002, pp.1~14.

[16] Robert, A., “gSOAP: Generator Tools for Coding SOAP/XML Web Service and Client Applications in C and C++”, http://www.cs.fsu.edu/~engelen/soap.html/.

[17] Apache Group, “Apache”, http://www.apache.org/.

[18] Apache XML project, “Axis”, http://xml.apache.org/axis/.

[19] Libxml, “The XML C parser and toolkit of Gnome”, http://xmlsoft.org/.

**최 현 미**

2002 홍익대학교, 컴퓨터공학과 학사

2002 ~ 2004 포항공과대학교, 정보통신학과 석사 과정

<관심분야> XML 기반의 네트워크 관리, Web Services.

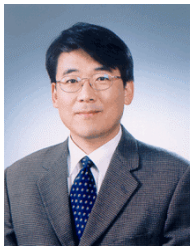
**최 미 정**

1998 이화여자대학교, 컴퓨터공학과 학사

1998 ~ 2000 포항공과대학교, 컴퓨터공학과 석사

2000 ~ 현재 포항공과대학교, 컴퓨터공학과 박사 과정

<관심분야> XML 기반의 네트워크 관리, 에이전트 기술, 정책 기반의 네트워크 관리



**홍 원 기**

1983 Univ. of Western Ontario, BSc in Computer Science

1985 Univ. of Western Ontario, MS in Computer Science

1985 ~ 1986 Univ. of Western Ontario, Lecturer

1986 ~ 1991 Univ. of Waterloo, PhD in Computer Science

1991 ~ 1992 Univ. of Waterloo, Post-Doc Fellow

1992 ~ 1995 Univ. of Western Ontario, 연구교수

1995 ~ 현재 포항공과대학교 컴퓨터공학과 부교수
2003 ~ 현재 한국통신학회 통신망운용관리연구회 위원장

2003 ~ 현재 IEEE ComSoc CNOM Vice Chair

<관심분야> 네트워크 트래픽 모니터링, 네트워크 및 시스템 관리, Network Security.

**박 용 석**

1986 서울대학교, 전자공학과 학사

1988 서울대학교, 전자공학과 석사

1996 Purdue Univ., PhD in Electrical and Computer Engr.

1996 ~ 1996 IBM, Research Scientist

1996 ~ 2000 AT&T, Senior Member of Technical Staff

2000 ~ 2001 Coree Networks, Systems Engineer

2001 ~ 2001 Lucent Technology, Systems Engineer

2002 ~ 현재 삼성전자 통신연구소, 수석연구원

<관심분야> 네트워크 장비개발, Network Management, Telecommunications Architecture and Services