석 사 학 위 논 문

NETCONF와 웹 서비스를 이용한 확장된 구성 관리 시스템

김 동 현 (金 東 鉉)

컴퓨터공학과 (네트워크 전공)

포항공과대학교 컴퓨터공학과

2005

NETCONF와 웹 서비스를 이용한 확장된 구성 관리 시스템

Extended network configuration management system based on NETCONF and Webservices

Extended network configuration management system based on NETCONF and Webservices

by

Dong-Hyun Kim

Department of Electronical and Computer Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Engineering in the Department of Electornical and Computer Engineering.

Pohang, Korea
December 21, 2004
Approved by

_____

Major Advisor: James Won-Ki Hong

Extended network configuration management system

based on NETCONF and Webservices

김 동 현

위 논문은 포항공과대학교 컴퓨터 공학과 석사 학위 논문으로 학위논문 심사위원회를 통과하였음을 인정함.

2004년 12월 21일

학위논문심사 위원회 위원장 홍원기 (인)

위 원 김 종 (인)

위 원 서영주 (인)

MCC
20022623

김 동 현, Dong-Hyun Kim, Extended network configuration management system based on NETCONF and Webservices, NETCONF와 웹 서비스 이용한 확장된 구성 관리 시스템, Department of Electrical and Computer Engineering, 2005, 57P, Advisor: J. Won-Ki Hong, Text in Korean.

ABSTRACT

인터넷은 다양한 벤더의 네트워크 장치들 간의 연결로 구성되어 있다. 이러한 환경 속에서 대다수의 네트워크 장치를 관리하기 위한 구성 관리(Configuration Management)에 있어서 취약점이 지적 되어져 왔다. 이 구성 관리의 문제점을 해결하기 위한 방법의 하나로 XML(Extensible Markup Language) 기반의 기술들이 논의되었다. 그 유용성이 표준화 기관들과 벤더들을 통해 논의 되고 증명되었었으나 이 과정에서 제약 없이 생겨나는 통신 모델, 정보 모델을 표준화에 의하여 통제될 필요성이 제기 되었다. 그 활동의 하나로 IETF(Internet Engineering Task Force)에서 Network Configure(NETCONF) Working Group을 통해 표준화 작업이 진행 되고 있다.

본 논문에서 우리는 NETCONF에서 진행 중인 인터넷 초안을 분석하고 문제점을 지적하고, 그에 대한 구현을 통한 해결 방안을 제시하고자 한다. 그리고 이러한 문제점을 보완한 XML기반의 구성 관리 시스템(XCMS-WS)을 제안한다. 우리가 구현한 XCMS-WS 시스템은 XPath를 사용한 구성 정보의 효율적인 조작 및 NETCONF 프로토콜을 이용하여 구성 정보의 효과적이며 표준화된 통신이 가능하며 Private UDDI를 이용한 에이젼트 관리 또한 효율적이다. 아직 표준화 단계에 머물러 있는 NETCONF 프로토콜의 실제 구현의 사례가 되고 웹 서비스 기술의 보다 현실적인 적용 사례가 될 것 이다.

목 차

그림 목차

표 목차

1. 서론

인터넷은 다양한 벤더의 네트워크 장치들 간의 연결로 구성되어 있다. 이러한 환경 속에서 대다수의 네트워크 장치를 관리하기 위한 구성 관리(Configuration Management)에 있어서 취약점이 지적 되어져 왔다. 이 구성 관리의 문제점을 해결하기 위한 방법의 하나로 XML(Extensible Markup Language)[1]기반의 기술들이 논의되었다. 그 유용성이 표준화 기관들과 벤더들을 통해 논의 되고 증명되었었으나 이 과정에서 제약 없이 생겨나는 통신 모델, 정보 모델을 표준화에 의하여 통제될 필요성이 제기 되었다. 그 활동의 하나로 IETF(Internet Engineering Task Force)[2]에서 Network Configure(NETCONF) Working Group[3]을 통해 표준화 작업이 진행 되고 있다.

NETCONF는 네트워크 관리에 있어서 크게 두 가지 부분에서 사용되는데 하나는 정보 모델 즉, 복잡한 관리 정보의 모델링하는 것이고, 또 다른 하나는 매니져와 에이전트간의 통신 모델 즉 메시지 포맷인 관리 프로토콜을 정의 하는 것이다. 이와 같이 정보 모델과 통신 모델의 명확한 분리도 NETCONF Protocol[4][5]을 사용한 장점이다.

NETCONF의 관리 프로토콜 메시지는 요청한 서비스를 위해 수행해야 할 오퍼레이션을 바로 호출 할 수 RPC(Remote Procedure Call)방법을 XML 태그를 사용하여 정의한다. 즉 정의 된 오퍼레이션들의 이름을 XML 태그로 표현되어 있어 에이전트측에서는 이 메시지를 분석하여 오퍼레이션을 실행하고 결과를 XML형식의 메시지에 담아 보내게 된다. 이과 같은 방법을 통해 XML메시지로 RPC호출이 가능하게 된다. NETCONF에서는 RPC를 제공하기 위한 방법 중 하나로 W3C(World

Wide Web Consortium)에서 정의한 SOAP(Simple Object Access Protocol)[5] 메시지를 이용한다. SOAP에서는 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 이 기종의 시스템 간에도 쉽게 데이터를 교환할 수 있는 장점이 생긴다.

NETCONF는 최근 인터넷 초안(Internet Draft)를 발표[7][8]하였으며 여기에는 이전의 NETCONF프로토콜의 개선점 및 NETCONF 전송 계층의 하나인 NETCONF over SOAP[23]에 대한 정의가 새로 내려져 있다. 이를 위한 연구로서 오픈 소스 진영에서는 MEDEYNES를 필두로 YENCA 프로잭트[24]가 진행되고 있다. 하지만 이는 NETCONF 프로토콜에 대한 구현일 뿐이며 실제 NETCONF Over SOAP, SOAP Over HTTP[23]에 대한 고려는 되어 있지 않다.

NETCONF 관련 인터넷 초안 및 그 관련된 시스템을 분석한 결과 우리는 3가지의 문제점을 발견했다. 첫째로 NETCONF 프로토콜에서 구성 정보 수정을 위한 메시지 구조가 addressing에 있어서 비효율적이며 복잡한 구성 정보 수정 시 여러 번의 메시지 전송이 필요하다는 점, 둘째로 SOAP의 유용성에 대해 설명하고 있으나 SOAP Binding에 대한 실제 레퍼런스 구현이 존재하지 않는다는 점, 마지막으로 WSDL, SOAP등의 웹 서비스 기술을 사용하나 이는 메시지 전송만이 고려 되어져 있어서 구성 장비 에이젼트의 발견과 호출에 대한 방법이 제시되어 있는 않는 점이 문제이다.

우리는 기존의 XML 관련 툴 및 웹 서비스[9]관련 기술들을 충분히 활용하여 매니저와 에이전트의 통신 수단으로 SOAP Over HTTP를 이용하고 있다. 이는 NETCONF 에이젼트를 웹 서비스화 시킬 수 있는 기반이 된다. 본 논문에서는 실제 서비스화 된 NETCONF 에이젼트를 UDDI[10]에 등록하여 서비스로 이용하는 구성 관리 시스템을 제시한다.

우리가 제시하는 XCMS-WS(XML-Based Configuration Management System using WebServices) 시스템은 XPath[11]를 사용한 구성 정보의 효율적인 조작 및 NETCONF 프로토콜을 이용하여 구성 정보의 효과적 통신이 가능하며 Private UDDI를 이용한 에이젼트관리 또한 효율적이다. 아직 표준화 단계에 머물러 있는 NETCONF 프로토콜의 실제 구현의 사례가 되고 웹 서비스 기술의 보다 현실 적인 적용 사례가 될 것 이다.

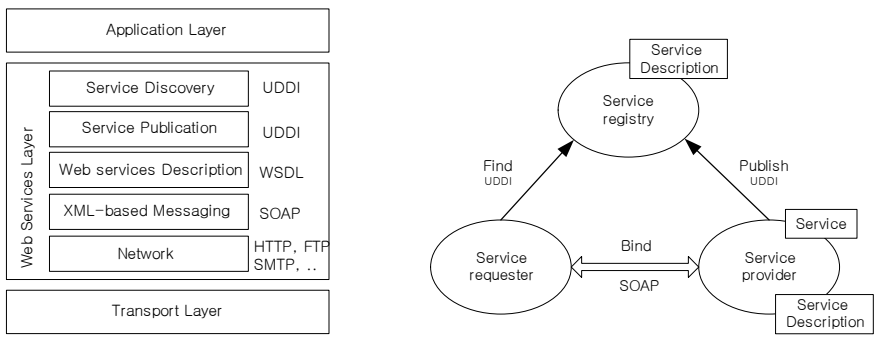
2. 관련 연구

이 장에서는 본 연구의 결과로 구현된 XCMS-WS 시스템에 적용되는 관련 연구로 웹 서비스 기술과 웹 서비스 기반의 네트워크 관리 시스템을 간략하게 알아 본다. 구성 관리의 선행 연구로서 Cisco나 Juniper Networks 같은 장비 업체에서 개발된 XML 기반 구성 관리 시스템[12][13]에 대해서 알아 보고 이의 문제점을 해결하기 위한 IETF에서 표준화 진행 중인 NETCONF 프로토콜을 소개한다.

2.1. 웹 서비스 기술 [14]

웹 서비스 아키텍쳐는 인터넷을 통해서 애플리케이션간의 통신을 위해

W3C에 의해서 정의되었다. 웹 서비스는 인터페이스와 바인딩을 URI(Uniform Resource Identifiers)[15]로 정의하고 XML로서 구현한다. 웹 서비스는 WSDL(Web Services Description Language)[16]을 통해 외부에 공개 되고 발견될 수 있으며 인터넷 프로토콜 위에 XML메시지(SOAP)로서 상호 작용이 가능하도록 한다. 플랫폼, 언어에 관련 없는 표준 웹 기술에 의한 컴포넌트간의 접근이 가능하다.



(a) Web services protocol, (programming) stack (b) Web services actors, objects, and operations

그림 1 : 웹 서비스 구조

웹 서비스는 전송 계층과 애플리케이션 계층 사이에 위치한다. 웹 서비스 레이어 자체는 그림 1(a)에 표현 한 것과 같이 WSDL, SOAP, UDDI, HTTP등 여러 인터넷 표준 프로토콜에 기초한다.

이중 상호작용을 위한 첫 번째 레이어를 구성하는 HTTP프로토콜은 범용적으로 널리 쓰이는 특징을 가지는 웹 서비스를 위한 가장 적합한 트랜스 포트 계층이다. 물론 인터넷에서 통용되는 도메인을 위한 SMTP[17], MIME[18], FTP[19]등과 인트라넷에서 사용되는 도메인을 위한 CORBA[37], Message Queue[38]등의 다른 트랜스 포트 계층도 사용이 가능하다. 두 번째 레이어는 SOAP으로 HTTP와 함께 HTTP

Post방식에 의한 페이로드의 전송으로 RPC호출을 한다. 이때 세 번째 레이어인 WSDL 문서는 서비스의 명세인 웹 서비스의 공용 인터페이스의 정의를 제공한다. 마지막으로 UDDI는 이 공용 인터페이스 정의를 공개 장소에 등록 가능하게 하며 이의 검색을 통해 공개된 서비스의 접근을 위한 방법을 제공한다.

그림 1(b)은 웹 서비스의 구성 요소간의 역할을 보여준다. 웹 서비스는 레이어 별로의 구분 이외에 역할에 따라 3가지로 나눌 수 있다. 서비스 제공자, 서비스 요청자, 그리고 서비스 레지스트리이다.. 아래에서 각각의 역할을 설명한다.

1. 서비스 제공자는 웹 서비스를 생성하고 서비스의 정의를 하고 서비스를 UDDI에 기반한 레지스트리에 등록한다. 웹 서비스가 일단 등록되면, 서비스 요청자는 UDDI 인터페이스를 통해 서비스를 발견한다.

2. 서비스 요청자는 UDDI를 통해 발견된 WSDL 서비스 디스크립션의 URL을 통해 서비스의 명세를 얻으며 이를 통해 서비스 요청을 한다.

3. UDDI 서비스 레지스트리는 서비스 요청자가 서비스에 직접 바인드 되기 위한 그리고 호출하기 위한 정보를 제공한다.

2.2. 웹 서비스 기반 네트워크 관리[20]

인터넷을 통한 매니지먼트 방법들이 다양하게 나타났지만 분산된 광범위한 서비스를 지원하는 표준화된 방법을 제공한다는 점에서 웹

서비스는 그 중에서 가장 중요한 하나가 될 듯하다.

실제 웹 서비스는 미래의 오퍼레이팅 시스템의 표준화된 구성 요소를 위한 기술이 될 것이며 이점은 네트워크, 애플리케이션, 시스템 모두에 걸쳐서 제공 되어지는 단일화된 통신 모듈을 제공할 수 있다. 웹 서비스의 상호 운용성의 강점은 네트워크 관리 분야에 있어서도 큰 강점으로 적용될 수 있다.

웹 서비스는 매니지먼트만을 위해 생겨난 기술이 아니어서 네트워크 관리를 위한 데이터 모델, 오퍼레이션 정의 등의 표준화의 필요성이 많은 연구가 필요한 분야이다.

2.3. Configuaiton management의 소개

XML을 이용한 네트워크 관리를 위한 여러 시도들이 있어 왔고 이는 구성 정보 관리에 있어서도 예외는 아니다. 네트워크 관리에서 구성 관리는. 중앙에서 관리하는 데이터 베이스, 구성 관리 패치 파일 등의 완벽한 동기화를 위한 관리 모델이 필요하다. 또한 벤더 종속적이지 않는 구조를 위한 XML 스키마의 정의도 논의 대상이다. 특히 XML을 기반으로 한 RPC는 장비들과의 데이터 교환을 위한 간단하고 호환성있는 기술을 제공한다.

Juniper의 JUNOscript[12]와 CISCO의 Configuration Registra[13]등이 이러한 방법론의 예이다. Juniper Networks' JUNOScript는 JUNOS NOS(Network Operating System)에서 시스템의 조작을 위한 스크립트로써 처음 소개 되었으며, XML을 기반으로 한 네트워크 장비의 관리 노력과 비용을 줄일 수 있는 간단하고도 효율적인 모델을 제시한다. 핵심인 JUNOScript는 클라이언트의 XML-RPC를 통한 구정 정보로의 접근과

수행을 가능하게 한다. JUNOScript Server는 태그의 분석을 통한 요청을 적절한 장비에 배분하는 역할을 하며 이 결과를 요청자에게 돌려준다.

CISCO의 Configuration Registrar 는 CISCO IOS 네트워크 장비로 구성 파일을 자동으로 배포 시키기 위한 웹 기반 시스템이다. 각 시스템에 위치한 시스코 네트워킹 서비스(CNS) 에이젼트를 동작시킨다. 장비가 처음 스타트 되었을 때 초기 구성 정보를 장비에 전송하고 이를 관리한다. 이는 HTTP로 XML 메시지를 에이젼트와 통신하는 방식으로 동작된다.

위에 설명한 두 시스템을 비롯한 상용 구성 관리 시스템은 관리에 있어서 실질적인 편리함을 제공한다. 하지만 이와 같은 구성 관리 방법은 앞에서 설명된 바와 같이 벤더 종속적인 데이터 구조와 통신 모델을 가진다.

이를 해결하기 위해 IETF에서는 시스코, 쥬니퍼등의 주요 벤더의 구성관리 시스템을 모델로 하여 NETCONF WG을 통해 구성 관리 모델의 표준화 작업을 하고 있다.

2.4. NETCONF의 개요

이 절 에서는 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 인터넷 초안에 대해 간략히 요약한다

NETCONF IETF Working Group의 NETCONF프로토콜은 JunoScript를 기반으로한 XML기반 구성 관리 프로토콜로써, 쥬니퍼와 시스코의 주도하에 이루어지고 있다. 프로토콜[4], 전송 방식[21][22][23], 아키텍쳐,

오토메이션등에 대한 표준화가 진행되어 지고 있다. INRIA (France), IUB (Germany), Postech (Korea)에서는 프로토타입의 개발이 시도되고 있으며, 프로토콜의 표준화가 완료되어 가는 현재까지 데이터 모델에 대한 표준화는 아직 이루어지지 않았다. 최근에 NETMOD[25]를 통해 데이터 모델의 표준화를 위한 활동이 시작되었다.

2.4.1. NETCONF 프로토콜

NETCONF는 매니져와 에이젼트 간의 간소화된 RPC메커니즘 기반의 통신 기능을 제공한다. 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 에이젼트에게 구성 정보를 가져오고, 수정하는 구성 관리 서비스를 RPC(Remote Procedure Call) 방법으로 제공하게 된다.

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. 이때 NETCONF Session을 통한 애플리케이션과 장치간의 논리적인 연결을 관리하기 때문에 하나 이상의 애플리케이션이 장비를 관리하는 경우 각 세션에 대한 관리 정보와 Global 구성 정보의 관리에 대한 메커니즘도 제공한다.

Configuration Model

NETCONF에서는 running, startup, candidate의 단계로 구성 정보의 상태가 존재하며, candidate에 구성 정보 변경의 히스토리를 저장해 둠으로써 구성 정보의 잘못된 수정에 따른 복구를 가능하게 하고 네트워크 장비의 잘못된 동작을 미연에 방지하고자 한다. 예를 들어 현재 장비가 수행되고 있는 구성 정보가 'running' 이며, 이때 구성 정보를 수정 시 각 수정 과정은 'candidate' 상태로 저장 되며 'candidate' 상태의 구성 정보 내용의 안정성이 검증(commit) 되면 'startup'이라는 이름으로 현 구성 정보를 저장하여 장비의 부팅 등의 재수행 시 구성 정보로 사용한다. 구성 정보 수정에 대한 오퍼레이션은 3장 시스템 설계에서 제시한다.

구성 모델과 각 단계의 설명은 아래와 같다.

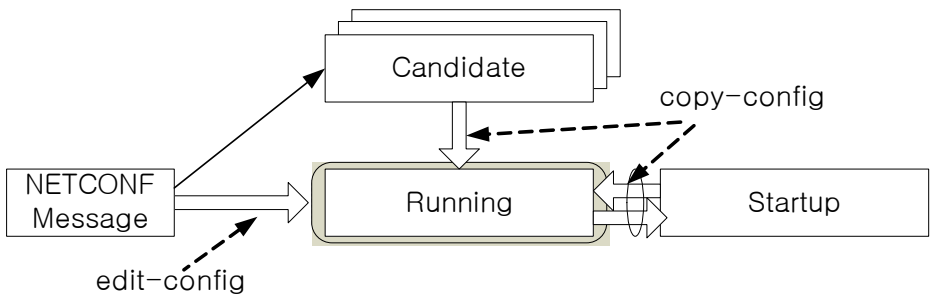


그림 2 : Configuration Model

- ✓ Candidate: 실제 네트워크 장비에 적용이 되지 않은 전체 구성 정보를 표현한 구성 정보이다. 구성 정보 변경의 히스토리에 해당하는 Data Store이며 실제 Running 상태로 변경할 수 있다.

- ✓ Running: 네트워크 장비의 동작 중인 구성 정보의 현재 상태를 나타낸다.

- ✓ Startup: Running 상태의 구성 정보는 자동으로 다음 번 재 시동 시에 적용되지 않는 단지 현재의 구성 상태일 뿐이다. 현재의 구성 정보를 다음 번 재 시동 시에도 적용되게 하려면 startup data store에 저장이 필요하다. 이는 copy-config 오퍼레이션을 통한 startup data store의 UPDATE로써 가능하다.

구성 관리를 수행하고자 하는 네트워크 장비의 전송, 관리 데이터, 관리 오퍼레이션 등의 환경은 매우 다양하기 때문에 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

.

계층	내용
Application	메시지들을 전달하는 전송 프로토콜

	BEEP, SSH, HTTP etc.
RPC	RPC 오퍼레이션에 대한 요청과 응답을 나타냄 <rpc>, <rpc-reply>, <rpc-error> etc.
Operation	NETCONF 에서 정의한 RPC 를 제공할 오퍼레이션들을 나타냄 <get-config>,<edit-config>,<copy-config>, etc.
Content	XML 형태의 관리 구성 정보를 나타냄 Configuration data

표 1 : NETCONF 관리 프로토콜의 계층적 구조

표 1은 NETCONF에서 정의하고 있는 4가지 계층과 그 계층에 포함되는 내용들을 예로 보여주고 있다.

APPLICATION LAYER

NETCONF는 계층으로 나누어 정의하여 전송 프로토콜에 대해서는 중립적이다. 따라서 어떠한 전송 프로토콜도 지원하는 것이 기본 개념이다. 하지만 현재는 SSH, 지원 프로토콜로써 BEEP, SOAP/HTTP에 대해서만 고려한다. APPLICATION 계층은 매니저와 에이전트의 연결(connection)에 대해 정의한다.

매니저와 에이전트가 하나의 세션(session)을 형성하면 NETCONF에서는 다중의 TCP 연결이 가능하다. NETCONF에서 이런 TCP 연결을 session이라 부르고 있다. 새 ID에서는 예전의 세 가지 종류의 채널 중 두가지 채널(Management, Notification channels)의 정의가 프로토콜의 간소화를 위해 사라졌다.

그림 3은 edit-config 오퍼레이션의 요청을 표현하는 NETCONF메시지의 예로써 NETCONF에서 정의하고 있는 4가지 계층

중 Application 계층을 제외한 나머지 3계층과 메시지와의 관계를 보여준다. 이 메시지는 실제 동작하는 장비의 상태 구성 정보인 running의 지정된 인터페이스 카드와 NETMASK의 구성 정보 값을 수정하게 된다.

RPC LAYER

NETCONF 프로토콜은 RPC기반의 통신 모델을 사용한다. 애플리케이션 프로토콜에 종속되지 않는 요청과 응답의 구조를 제공하기 위해 <rpc>, <rpc-reply>를 사용한다. 또한 매니저가 요청한 메시지에 대해 에이젼트가 제어할 수 있는 방법을 <rpc> 태그 안에 요청 메시지를 명시할 수 있는 'message-id'를 정의를 통해 제공하게 되는데 이는 메시지 번호에 의해 이전에 요청한 메시지를 제어 하는 핸들러로써 사용된다. 즉 message-id는 매니져와 에이젼트의 접속에 의해 Session이 이루어 진 후 세션을 통해 주고 받는 메시지를 관리하기 위한 번호라고 할 수 있다. 그림 3에서의 RPC태그가 RPC 레이어를 보여준다. attribute의 message-id의 번호 107은 메시지가 매니져와 에이젼트 사이의 세션에서 107번째의 NETCONF 프로토콜 메시지라는 의미이다. 또 RPC 레이어에는 Operation인 edit-config가 하위 항목임을 볼 수 있다. 이는 RPC를 통해 수행되는 오퍼레이션이 edit-config 오퍼레이션임을 표시하는 것이다.

OPERATION LAYER

NETCONF는 SNMP에서 제공하고 있는 'get', 'set', 'trap'에 대응할 수

있는 get-config, edit-config, notify 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 잠금(lock)을 걸 수 있는 오퍼레이션이 정의된다. 이는 SNMP에서 제공하지 않는 관리 오퍼레이션들이 새롭게 정의 되었음을 알 수 있다. 아래에는 retrieve, configure, copy, delete에 해당하는 NETCONF 프로토콜의 기본 오퍼레이션을 보여준다.

<get-config>: 에이전트가 관리하고 있는 구성정보의 특정 부분을 또는 전체를 불러 올 때 사용하는 오퍼레이션이다. 즉 에이젼트가 가지고 있는 구성 정보를 매니저에게 전송하고자 할 때 호출 된다.

<edit-config>: 실제 구성 관리의 실질적인 일을 하는 오퍼레이션이다. 에이전트가 갖고 있는 구성 정보를 변경할 때 수행되는 세 가지 오퍼레이션을 가진다. merge는 새로운 구성 정보를 추가할 때, replace는 기존 구성 정보를 변경 할 때, delete는 기존 구성 정보를 삭제 할 때 호출 된다.

<copy-config> : 매니저가 에이전트에게 구성 정보를 원격으로 로딩하거나 에이젼트의 Running Data Store를 Startup Data Store로 아니면 그 반대의 경우를 수행할 수 있는 오퍼레이션이다. copy-config가 구성 관리 정보의 일부분을 조작한다면 이 오퍼레이션은 전체의 백업 및 리스토어가 가능하게

한다.

<delete-config> : Running 이나 Startup, Candidate Data Store의 삭제를 하는 오퍼레이션이다. 물론 이를 위하여 Data Store가 락 이 걸려 있지 않음을 전제로 한다.

그림 3 메시지에서 Operation의 실제 사용을 보여준다. edit-config의 Operation 에 필요한 파라미터인 target와 config정보가 메시지에 포함되어 있으며 target의 running은 실제 수정될 구성 정보 저장소이며 config에는 교체될 구성정보가 포함된다.

CONTENTS LAYER

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 NETMOD를 통해 진행하고 있다. 그림 3에 Content 레이어의 실제 구조를 볼 수 있다. config 태그의 아래 부분이 Content 레이어이며 포함된 내용이 교체 정보임을 표시하는 replace 값을 가진 operation attribute와 실제 교체 정보가 포함된다. 이때 name와 mtu의 값으로 어떤 interface의 값이 교체될 것인지를 addressing한다.

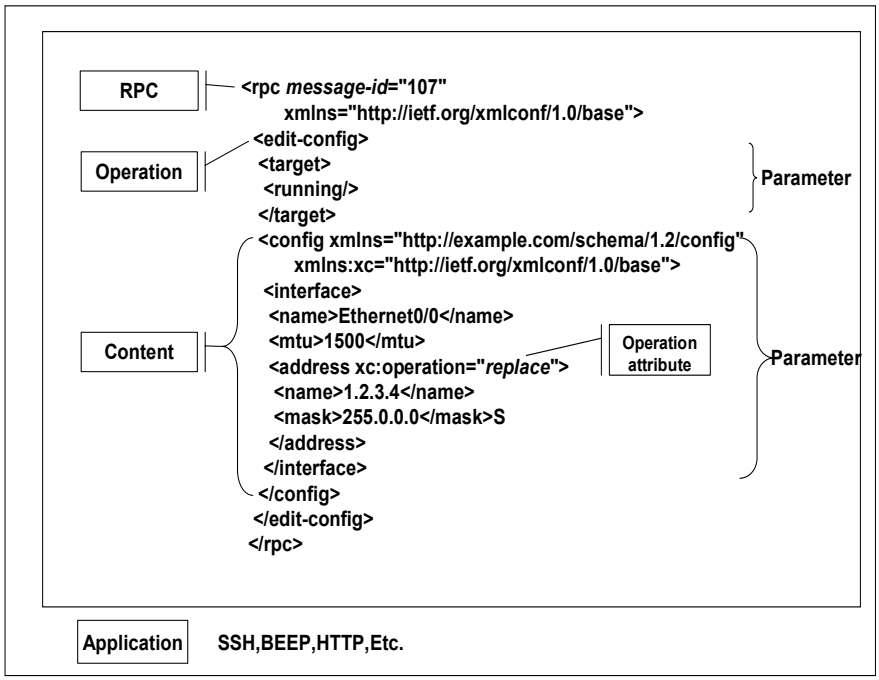


그림 3 : Example of NETCONF Protocol Message for <edit-config>

2.4.2. NETCONF 의 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 인터넷 초안을 제안하고 있다. 이 장에서는 각 인터넷 초안을 비교, 분석하여 각 전송 프로토콜의 장단점을 요약 정리하고자 한다.

SSH(Secure Shell)[21]는 기존의 네트워크 장비의 관리에 있어서 주로 사용해 오던 환경이다. 따라서 기본적으로 네트워크 오퍼레이터들의 강력한 요구에 의해 제공되어야 하는 전송

프로토콜이다. 그러나 SSH의 인터넷 초안은 구현의 간편성을 위하여 관리 수행을 위한 하나의 TCP연결만을 제안하고 있다. SSH의 경우 에이전트가 매니저에게 메시지를 비 동기적으로 보낼 수 있는 방법이 제공되지 않기 때문에 보고 채널 같은 경우는 유지할 수도 없다.

BEEP(Block Extensible Exchange Protocol)[22] 은 현재 Cisco와 Juniper Networks와 같은 대표적인 네트워크 장비 업체에서 추천하고 있는 전송 프로토콜이다. BEEP는 SSH와 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 비 동기적 메시지를 쉽게 처리할 수 있는 장점이 있다.

SOAP over HTTP 에서의 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 비 동기적으로 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. 그림 4는 Application계층으로 SOAP Over HTTP가 사용되었을 경우에 메시지의 구조이다. HTTP Header에는 Content-Type이 XML이며 , User-Agent의 값을 통해 Axis를 사용하는 것을 알 수 있다. HTTP Payload로 SOAP메시지가 포함되며 SOAP 프로코콜의 Payload로 NETCONF 프로토콜의 메시지를 포함한다.

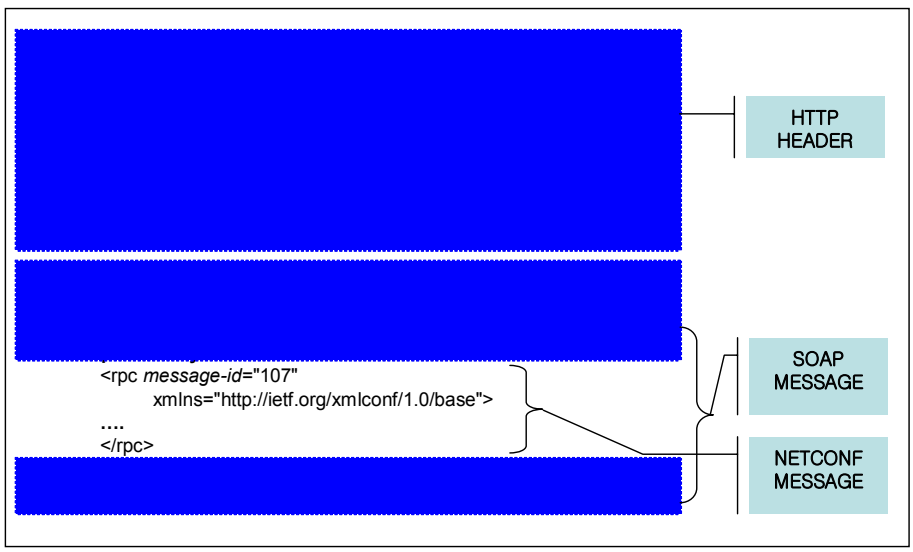


그림 4 : SOAP Over HTTP 방식을 통한 NETCONF 메시지

위의 세 전송 프로토콜을 표 2에 정리해 보았다. SSH는 에이전트에 직접 연결해서 매니징 하는 CLI환경을 고려하여 제안되고 있고, 매니저와 에이전트간의 통신을 위해서 현재 BEEP, SOAP over HTTP 두 개

를 놓고 한창 논의가 이루어 지고 있습니다. 멀티 채널을 지원하고 있는 BEEP이 있지만, 새 인터넷 초안에서는 Management, Notification 채널이 사라 짐으로써 SOAP Over HTTP의 장점이 더 부각 되고 있습니다. 본 논문에서는 RPC interface를 제공하고, Independent한 transport를 제공하고 있는 SOAP의 장점을 고려하고, XCMS-WS 시스템을 구성 관리 뿐만 아니라 웹 서비스 기반의 통합 네트워크 관리 시스템으로의 확장을 고려하여 SOAP/HTTP방식을 사용한다.

	SSH	BEEP	SOAP over HTTP
Operation Channel	support	Support	support
RPC Interface	Not	Not	Support
Independent-transport	Not	Not	Support
Note	Server/ Client	peer-to-peer	Server/Client

표 2 : 프로토콜 전송방식 비교

2.5. NETCONF 시스템 비교

본 절에서는 NETCONF 프로토콜의 표준안에 따라 현재 구현된 구성 관리 시스템 두 가지를 알아본다. 본 논문의 이전 연구였던 XCMS 시스템과 오픈 소소 프로잭트인 YENCA 이다. 이들의 차이점을 비교해 보고 각 시스템에서의 개선점을 알아 본다.

2.5.1. XCMS 시스템

XCMS(XML Based Configuration Management System)[26] 는 본 연구의 이전 연구이다. POSTECH, DPNM 연구실에서 진행한 NETCONF 프로토콜에 대한 연구로써 2003년 11월까지의 표준을 참조하여 제작하였다. 또 표준의 개선점을 찾아 표준을 수정하고 구성 관리의 효율성을 꽤하고자 하였다.

특징은 XPath로서 구성 정보를 포인팅한다. NETCONF 프로토콜의 RPC 계층을 SOAP메시지에 통합하고, 아직 표준에서 고려되지 않았던 XPath모드를 통해 관리 정보를 전체가 아닌 일부만 가지고 조작이 가능하게 하여 NETCONF 메시지의 효율성을 높였고, 아직까지 실제 구현에 대한 예가 존재하지 않던 NETCONF Over SOAP, SOAP Over HTTP를 통한 NETCONF 프로토콜을 보여준다.

하지만 최신 표준의 지원이 되지 않는 점과 표준에 부합되지 않은 오퍼레이션의 설계, RPC 레이어의 SOAP메시지에 통합에 의해 관리 정보와 통신 정보의 분리가 되지 않는 점은 개선이 필요하다.

2.5.2. YENCA

NETCONF 프로토콜은 표준화와 함께 구현에 대한 연구도 함께 이루어지고 있다. 이들 연구 중 OPEN Source Project인 YENCA는 프랑스의 MEDYNES[24]에서 진행 중 이다. 에이젼트의 디바이스 및 구성 요소의 모듈화가 특징이며 NETCONF 프로토콜의 특징인 표준화된 메시지를 잘 활용한 디바이스의 관리 구조는 이 시스템의 장점이라 할 수 있다. 모듈에 단일화된 인터페이스를 제공함으로써 확장성 측면에서 좋다. 하지만 TCP 커낵션을 통한 NETCONF 프로토콜 메시지

의 처리만 고려 된 구현이어서 SOAP전송, BEEP, SSH등의 표준 전송 방법에 대한 레퍼런스 구현이 필요하다.

	XCMS	YENCA
특징	- XPath모드를 이용한 NETCONF 프로토콜 메시지의 단순화	- 기본 프로토콜 구현의 충실 - 모듈 구조로 인한 구성 정보의 확장성
개선점	- 최신 표준의 지원이 되지 않음 - 표준에 부합되지 않은 오퍼레이션의 설계 - 관리 정보와 통신 정보의 분리가 되지 않음	- 단순 프로토콜의 구현 - SOAP전송, BEEP, SSH등의 전송 프로토콜 사용 안 함

표 3 : NETCONF 시스템 장단점 비교

본 연구의 구현물인 XCMS-WS에서는 위의 두 가지 시스템의 특징을 반영하고 NETCONF의 최신 표준을 따르도록 설계 한다. XCMS-WS만의 개선 사항도 포함한다. 개선 사항은 3장에서 설명한다.

3. XCMS-WS의 설계

이번 장에서는 구성 관리를 위한 표준화 프로토콜인 NETCONF의 분석에서 나타난 문제점을 지적하고 해결하기 위한 방안을 제시한다.

3.1. XCMS-WS의 Requirement

NETCONF 인터넷 초안 및 일반적인 구성 관리 시스템에서 요구되는 기본 요구 사항과 본 논문에서 제시하는 XCMS-WS시스템에서의 확장된 요구 사항을 알아 본다.

3.1.1. 기본 요구 사항

XCMS-WS에 포함된 기존의 NETCONF 인터넷 초안 및 일반적인 구성 관리 시스템에서 요구되는 사항은 다음과 같다.

(1) 매니져가 에이젼트에게 구성 정보를 업 로딩을 하거나 다운 로딩이 가능해야 한다. 매니져가 관리 하고 있는 모든 에이젼트의 구성 정보를 중앙에서 일괄적으로 처리 할 수 있는 구조를 제공한다.

(2) CLI 를 제공하여야 한다.

(3) 하나의 메니져에 여러대의 에이젼트를 관리 할수 있어야 한다.

(4) 구성정보에 잠금을 할 수 있어야 한다. 이는 여러 매니져가 동시에 같은 구성 정보를 수정하려고 할 때 발생되는 동기화의 문제점을 해결해 준다.

(5) 매니져와 에이젼트의 상호 작용을 위한 XML 형식의 메시지 구조를 정의한다.

(6) 매니져와 에이젼트의 쉬운 개발 환경을 제공하기 위해 기존의

XML 관련 툴들이나 구현된 소스들을 활용하여야 한다.

3.1.2 확장된 요구 사항

XCMS-WS 시스템에서 새롭게 제시하는 요구 사항은 다음과 같다.

(1) XPATH 를 적용하여 구성 정보 모델을 수정이 가능하여야 한다.

(2) 확장된 XPath 사용과 동시에 NETCONF 표준이 사용가능해야 한다.

(3) XPath 를 이용한 복잡한 구성정보 수정을 위한 메시지 표현 가능해야 한다

(4) 웹 서비스를 이용한 메시지 전송 및 관리 구조를 제공한다.

(5) SOAP Binding 을 이용한 메시지 전송이어야 한다.

(6) UDDI 적용을 통한 구성 장비의 발견이 가능해야 한다.

(7) 발견한 에이젼트의 WSDL 을 통한 동적 호출이 가능해야 한다.

NETCONF ENABLE된 장비 발견 및 관리 구조는 UDDI를 통해 알 수 있고 에이젼트는 모듈화 관련 아키택쳐를 통한 관리 객체의 확장이 가능하다. 디바이스의 세부 구성 정보는 NETCONF의 초기 메시지를 통해 상호 교환이 가능함으로써 NETCONF 프로토콜과 UDDI를 통한 관리 구조의 결합은 구성 정보의 관리 시스템으로서 적합함을 보인다.

3.2. XCMS-WS 설계

XCMS-WS의 설계 과정은 앞서 제시한 시스템과 표준의 문제점을 반영하여 개선점을 제시한다. XPath를 이용한 오퍼레이션의 개선 방안과 SOAP, WSDL을 이용한 관리 정보 전송 방법과 UDDI를 이용한

개선된 매니지먼트 방법을 포함한다.

3.2.1. 오퍼레이션 및 메시지 구조 설계

[오퍼레이션의 문제점]

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. 예로서 <get-config> 오퍼레이션의 경우, 선택하고 싶은 구성 정보를 정확히 표현 할 수 없기 때문에 오퍼레이션 수행 결과가 선택되지 않은 구성 정보와 함께 보여주기도 한다.

[XPATH를 이용한 해결책]

이 문제점을 해결하기 위해서 우리는 XML 데이터의 노드의 위치를 명료하게 표현할 수 있는 표준 방법인 XPath의 사용을 제안한다. XPath를 사용하게 되면 <get-config>의 경우는 선택하고 싶은 구성 정보만을 가져올 수 있고 <edit-config>에서는 부분 수정이 가능하다.

XPath를 사용한 정의와 사용하지 않은 표준 프로토콜의 차이를 보이기 위해 예를 든다. 표 4는 XPath를 사용하지 않은 edit-config의 replace 명령이다. 변경할 구성 정보의 저장소를 선택하는 Target, 변경할 구성 정보를 지정하는 config가 존재하고 config에는 실제 변경이나 추가, 삭제 할 구성 정보임을 표시하기 위한 operation 애트리뷰트가 있다.

```
<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>
```

표 4 : XPath 를 사용하지 않은 <edit-config>의 replace

표 5는 XCMS에서 제시한 edit-config를 수정한 operation과 XPath를 이용한 방법이다. edit-config-{merge|replace|delete}와 같은 표현 방식으로 operation의 인지 단계부터 바로 서비스의 삭제, 추가, 변경 여부가 판단되는 구조이며 XPath를 통하여 선택되는 구성 정보의 직접 수정 작업을 가능하게 한 오퍼레이션 구조이다. 이 방법은 변경 전의 NETCONF 메시지에 비해 구성 정보의 일부분만을 지정하여 세밀한 조작이 가능하고 변경될 부분만 전달하는 방식으로 그 크기가 작아 지는 장점이 있다. 하지만 복잡한 구성 정보 변경을 위해서는 여러 번의 메시지의 전달이 필요하게 되어 트랜잭션에 대한 구현 및 정의가 필요하게 된다.

```
<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 5: XPath 를 사용한 <edit-config-replace>

최신 NETCONF 프로토콜[4] 관련 인터넷 초안에서는 XPath에 대한 정의가 추가 되었다. 여기서는 get-config시 XPath를 이용하여 구성 정보를 얻어 오는 방법을 제시하고 있다. 표 6에서와 같이 제시된 새로운 방법은 제시된 Filter 태그와 attribute인 type을 이용하여 XPath를 이용할 것임을 정의하고 가져올 구성 정보의 표현이 XPath로 표시된다. 새로운 인터넷 초안에서도 edit-config의 XPath의 방법은 제시되어 있지 않다. 표 6 예제의 내용은 running 구성 정보 저장소에서 user의 이름이 'fred'인 user의 전체 정보를 가져오게 하는 NETCONF 메시지이다.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath">
      users/user[name="fred"]
    </filter>
  </get-config>
  </rpc>
```

표 6 : XPath 를 사용한 <get-config>

새 인터넷 초안에서는 edit-config에서 XPath를 사용하진 않지만 edit-config에 대한 변경이 있었다. top태그 추가를 통해 구성 정보가 최상위 노드에 해당함을 표시하고 변경이 필요한 노드를 표시하는 방법이 변

경되었다. 표 7에서 나타나듯이 address 노드에 operation attribute가 지정되는 것이 아니라 그를 포함하는 interface 노드에 operation attribute가 추가 된다. 변경을 하게 되는 최상위 노드에 {replace|merge|delete}를 표현함으로써 구현시 하위 노드까지 해석하지 않아도 오퍼레이션의 판단이 가능하게 되었다. 하지만 이 방법 역시 replace시 변경이 되는 구성 정보의 노드 전체를 표현 해야 된다.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <top xmlns="http://example.com/schema/1.2/config">
        <interface xc:operation="replace">
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
          <address>
            <name>1.2.3.4</name>
            <mask>255.0.0.0</mask>
          </address>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>
```

표 7 : 변경된 <edit-config>의 replace 의 표준안

본 논문에서 제시하는 edit-config에서 XPath를 이용하는 방법은 edit-config 오퍼레이션에서 나타난 여러 문제들을 해결할 수 있는 방법을 제시한다.

XCMS시스템의 XPath방식에서의 문제점인 복잡한 구성 정보 수정의 표현 시 문제점을 config를 수정하여 해결하였으며 이 역시 XPath를 사용한다. 새로 제시한 방법은 표 8와 같다.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
          <target>
             <running/>
          </target>
          <configs>
               <config xpath =
“//interface/name[.="Ehernet0/0"]/following::mtu[.="1500"]/following::address” operation =
“replace”>
                    <address>
                         <name>1.2.3.4</name>
                         <mask>255.0.0.0</mask>
                     </address>
               </config>
               <config xpath =
“//interface/name[.="Ehernet0/1"]/following::mtu[.="1500"]/following::address” operation =
“delete”>
               <config xpath =
“//interface/name[.="Ehernet0/2"]/following::mtu[.="1500"]/following::address/mask” operation =
“merge”>
          </configs>
     </edit-config>
</rpc>
```

표 8 : XCMS-WS 에서 제시하는 XPath 를 사용한 <edit-config>

새 방법에서는 configs 테그를 추가 하였다. 그리고 하위 노드에는 config 노드가 변경 사항의 개수 만큼 추가된다. 각각의 config 노드에는 merge, replace, delete 를 위한 구성 정보가 xpath 에트리뷰트를 통해 포인팅 되고 하위 노드에는 교체되거나 추가될 구성 정보가 표현된다. 예를 들어 표 8 에서 configs 의 첫번째 config 노드를 보면 xpath 를 통해 Eternet0/0 이고 mtu 가 1500 인 interface 의 address 에 해당하는 구성 정보의 위치를 포인팅 하고 그 위치의 정보를 replace 하겠다는 의미이다. 따라 오는 “<address>……</address>”의 구성 정보로 변경된다.

이 방법을 통해 구성 정보의 일부분만을 지정하여 세밀한 조작이 가능하고 변경될 부분만 전달하는 방식의 XPath의 특징을 포함하면서 동

시에 복잡한 구성 정보 변경을 위해서는 여러 번의 메시지의 전달이 필요하게 되던 문제점을 해결할 수 있다. 또한 이 방법은 하나로서 처리되어야 할 명령이 나뉨으로 인한 트랜잭션에 처리에 대한 처리가 필요 없게 된다.

[XCMS-WS의 Operation의 정리]

XCMS-WS의 Operation 계층의 메시지는 파라메터에 해당하는 target, config, transaction등 각 오퍼레이션 마다 필요한 파라메터가 존재하고 이 파라메터가 NETCONF 프로토콜의 XML메시지에 표현 되어 진다. 위의 새 인터넷 초안과 XPath 표현식을 적용한 XCMS-WS의 Operation별 메시지는 표 9와 같다.

get-config	
<get-config> <source> <running/> </source> **<filter type="xpath">** users/user[name="fred"] **</filter>** </get-config>	Operation Name : get-config Param1 : source Param2 : filter Param3 : config
edit-config	
<edit-config> <target> <running/> </target> **<configs>** **<config xpath = "//interface/name[.="Ehernet0/0"]/following::mtu[.="1500"]/following::address" operation = "replace">** <address> <name>1.2.3.4</name> <mask>255.0.0.0</mask> </address> </config> **</configs>** </edit-config>	Operation Name : edit-config Param 1: target Param2 : configs

copy-config	
<get-config> <source> <running/> </source> <target> <startup/> </target> </get-config>	Operation Name : copy-config Param1: source Param2: target
delete-config	
<delete-config> <target> <startup/> </target> </delete-config>	Operation Name : delete-config Param 1: target

표 9 : XCMS-WS 의 Operation 표현

오퍼레이션 별 메시지와 오른쪽에는 각 오퍼레이션의 이름과 그 필수 파라메터를 명시하였다. 예로서 copy-config는 파라메터로서 source, target을 가지며 running, 즉 현재 동작 중인 장치의 구성 정보를 startup 데이터 저장소에 복사하게 된다. Startup이 변경 됨으로써 다음 번 장비 시동 시 지금의 구성 정보를 가지고 시작하게 된다.

XCMS-WS에서 NETCONF 프로토콜은 가장 중요한 오퍼레이션인 edit-config를 확장한 구조이다. XPath사용을 통해 이전 시스템 XCMS의 장점인 구성 정보 Addressing의 잇점을 가지고, 추가로 복잡한 구성 정보 수정 시 여러 번의 오퍼레이션을 내려야 하는 프로토콜이 비효율성을 해결 하였다. 즉 하나의 구성 정보 변경에 있어서 여러 번의 메시지 전송과 반응이 필요하던 문제를 해결하였다. 이로 인해 구성 관리 시스템의 평가 요소 중의 하나인 트랜잭션 처리에 있어서 안정성을 높일 수 있게 된다.

3.2.2. 서비스 호출 구조 설계

메시지를 전달함에 있어서 SOAP프로토콜을 사용하는데 이를 위해서는 매니져와 에이젼트간에 서비스의 명세에 대한 정보가 필요하다. 이를 위해 제공되는 것이 WSDL이다. WSDL은 에이젼트의 서비스 작성시 같이 만들어 지며 이는 SOAP 구현 환경에서 제공하는 툴을 이용하여 자동으로 생성된다. 그림 5은 에이젼트에서의 WSDL파일 자동 생성과 매니져와 에이전트 간의 서비스 호출 구조를 보여준다.

에이젼트에서는 구현된 서비스 코드의 헤더 파일을 통해 WSDL파일 및 서비스의 스켈레톤을 생성하고 매니저 측에서는 공개된 WSDL 문서를 통해 에이젼트의 서비스를 호출 할 수 있는 Proxy를 생성하게 된다. 이를 매니져 애플리케이션에서 사용함으로써 에이젼트와 매니져간의 RPC호출이 가능하다. WSDL파일의 portType의 Operation 이름을 가진 인터페이스를 생성하고, binding의 정보를 가지고 Proxy를 생성한다.

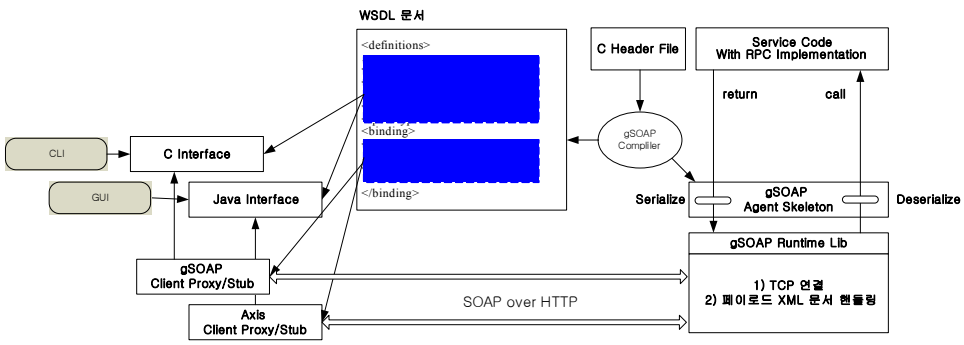


그림 5 : XCMS-WS 의 WSDL 이용한 서비스 호출 구조

표 10은 툴을 이용하여 코드로부터 자동 생성된 <rpc> 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.

```
<?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 10 : 오퍼레이션 방식의 XCMS 의 WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
  <definitions name="Netconfd" targetNamespace="urn:Netconfd" xmlns:tns="urn:Netconfd" xmlns:SOAP-
    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">
  <message name="netconfdRequest">
    <part name="req-msg" type="xsd:string" />
  </message>
  <message name="netconfdResponse">
    <part name="rpl-msg" type="xsd:string" />
  </message>
  <portType name="NetconfdPortType">
      <operation name="netconfd">
          <documentation>Service definition of function ns__netconfd</documentation>
          <input message="tns:netconfdRequest" />
          <output message="tns:netconfdResponse" />
      </operation>
  </portType>
  <binding name="Netconfd" type="tns:NetconfdPortType">
      <SOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
          <operation name="netconfd">
              <SOAP:operation soapAction="" />
              <input>
                  <SOAP:body use="encoded" namespace="urn:Netconfd"
                      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
              </input>
              <output>
                  <SOAP:body use="encoded" namespace="urn:Netconfd"
                      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
              </output>
          </operation>
    </binding>
  <service name="Netconfd">
          <documentation>gSOAP 2.7.0d generated service definition</documentation>
          <port name="Netconfd" binding="tns:Netconfd">
              <SOAP:address location="http://141.223.82.6:5455" />
          </port>
    </service>
</definitions>
```

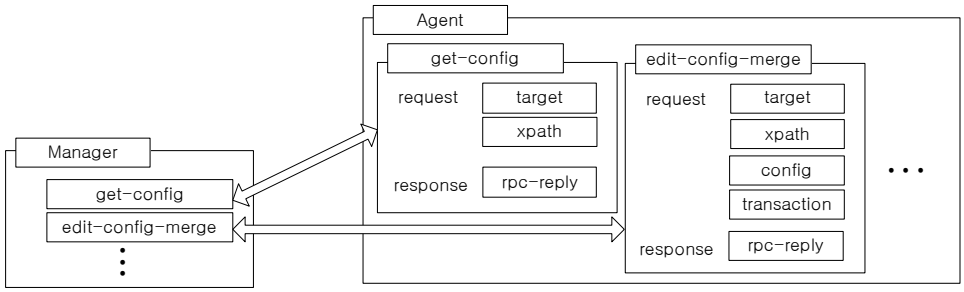
표 11 : 서비스 방식의 XCMS-WS 의 WSDL

XCMS-WS 시스템의 설계 시 서비스 호출에 있어서 오퍼레이션 방식과 서비스 방식을 고려하였다. 오퍼레이션 방식은 오퍼레이션 별로 서비스가 일대일로 대응된 웹 서비스를 공개하는 방식이다.

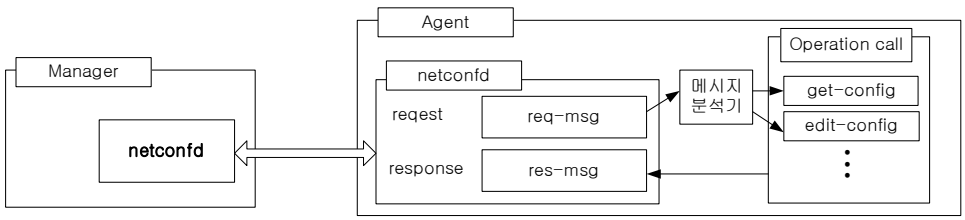
이 방식은 이전 시스템인 XCMS에서 채택한 바 있다. 이를 이용하면 각 오퍼레이션의 판단과 호출을 위한 매니저 측에 또 다른 하나의

계층이 필요하며 그림 6(a)에서 보여주듯 매니져 측에서 직접 호출을 하는 구조를 가진다. 오퍼레이션이 판단되는 시점이 매니져에서 이루어져야 하며 에이젼트는 get-config, edit-config등 각각의 오퍼레이션을 서비스로서 등록하는 과정이 필요하게 된다. get-config의 예를 들면 get-config에 해당하는 서비스가 에이젼트에서 등록되어 있고 source와 xpath의 파라메터 값과 함께 SOAP프로토콜을 통해 이를 호출하게 된다. 그리고 그 결과로써 rpc-reply의 값으로 결과를 리턴한다. 전달되는 get-config 오퍼레이션과 그 파라메터가 정의된 WSDL 문서는 표 10와 같다. 이 방식은 전달되는 메시지가 NETCONF 프로토콜 메시지가 아니다. 따라서 본 시스템의 설계에는 서비스 방식의 호출을 사용한다.

서비스 방식은 NETCONF 메시지를 생성하고 이를 에이젼트 측에서는 해석하여 필요한 오퍼레이션을 수행 하게 된다. 에이젼트 단위로 관리 정보를 가지게 되는 구성 관리 시스템 구조의 특성상 서비스 방식의 호출이 NETCONF 메시지를 전달하는 구조에 적합하다고 판단했기 때문이다. XCMS-WS의 서비스 호출 방식은 그림 6(b)에서 보여준다. 서비스 방식의 호출은 에이전트 자체를 웹 서비스화 한 netconfd 서비스를 호출한다. 표 11의 WSDL파일에서 보여 지듯 에이전트를 접근하고 그 결과를 되돌려 주기 위한 하나의 request와 response형 만 존재한다. 이 를 통해 NETCONF 메시지를 전달하고 에이젼트의 서비스 안에서 메시지 분석기를 통해 메시지를 해석하여 필요한 오퍼레이션을 재 호출하는 구조를 가진다.



(a) XCMS 오퍼레이션 방식



(b) XCMS-WS 서비스 방식

그림 6 : 서비스 방식과 오퍼레이션 방식의 비교

3.2.3. UDDI 를 이용한 에이전트 관리 구조 설계

[필요사항]

NETCONF 프로토콜은 구성 관리에 있어서 장비들의 구성 정보 변경을 수행하기 위하여 필요할 오퍼레이션을 정의 한 것이다. 이와 같이 구성 관리에 있어서는 실제 이들간의 주고 받는 메시지의 표준화도 중요하지만, 이 프로토콜을 어떻게 활용할 것인가에 대한 비젼, 즉 구성 장비의 관리를 위한 방법의 제시도 필요하다. NETCONF 프로토콜의 실제 프로토콜 수준의 구현이나 오픈 소스 등은 존재하나, 실제 사용 예는 없다. 특히 SOAP바인딩에 대한 실제 시스템의 적용 및 UDDI를 통한 에이전트의 관리에 대한 구성 관리 시스템의 전체

아키택쳐는 제시된 바가 없다. 본 논문에서는 프로토콜의 실제 사용이 가능한 구현을 통해 SOAP over HTTP의 유용성을 증명하고 이 매니져, 에이젼트 아키택쳐를 UDDI통해 확장한 구성 관리 시스템 아키택쳐를 제시한다.

[해결점]

웹 서비스라고 하면 WSDL을 이용한 서비스의 공표와 UDDI 를 통한 서비스의 발견, 또 SOAP을 이용한 원격 서비스의 호출을 이용한 구조가 일반적이다. NETCONF 에서 제시하는 프로토콜에서는 WSDL의 정의와 SOAP을 통한 메시지 전달에 대한 권고 사항이 정의되어 있다.

XCMS-WS에서는 이를 준수하고 추가로 UDDI를 이용한 확장을 제시한다. 여기서 쓰이게 되는 UDDI 구조를 보면 businessEntity, businessService, bindingTemplete, tModel 네가지 요소로 구성된다.

그림 7은 네가지 요소의 일반적 사용을 보여준다. businessEntity는 비즈니스와 기관을 표현한다. 이름, description, contact people 등의 엘로우 페이지가 여기에 해당하는 정보이다. businessService는 회사나 기관의 서비스의 high level description을 제공한다. 이 정보는 전화번호부의 화이트 페이지의 정보와 유사하다. bindingTemplete는 제시된 비즈니스 서비스의 technical description을 제공한다. 이는 웹서비스의 access point (예로 url이나 e-mail 주소)나 acesspoint에 이르게 하는 간접 방법을 지원한다. 마지막으로 tModel은 technical 모델로써 웹 서비스 개발자로부터 사용되는 기술 문서의 포인터를 포함하고 문서의 메타 데이터를 포함한다. 웹 서비스의 제공자와 사용자 사이에서 호환성을 확인하기 위하여 사용되어지기 위해 준비된다.

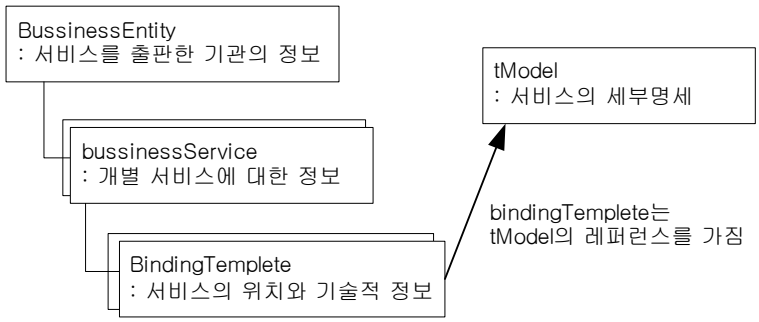


그림 7 : UDDI 레지스트리 구조

처음의 businessEntity, businessService, bindingTemplate 세 요소는 포함관계의 계층 구조이다, business entity는 하나 이상의 business service를 포함하고, businessService는 하나 이상의 bindingTempletes를 포함한다.

XCMS-WS에서 사용되는 에이젼트 관리는 비즈니스 엔티티를 사용한다. UDDI registry의 비즈니스 엔티티는 비즈니스 엔티티를 다른 회사에 제공하기 위해서 비즈니스 서비스를 알리는데 사용되어 진다. 우리는 이 비즈니스 엔티티를 웹 서비스를 이용한 네트워크 관리 요소들을 등록하는 곳에 사용한다. 표 12는 XCMS-WS의 에이젼트에 UDDI 레지스트리에 등록된 businessEntity 를 보여준다.

```
<?xml version="1.0" encoding="utf-8" ?>
  <businessEntity xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" businessKey="8fcc8ad7-e3e4-47b6-85e1-
5c97836e4b75" operator="Microsoft Corporation" authorizedName="KIM DONG HYUN" xmlns="urn:uddi-
org:api_v2">
  <discoveryURLs>
            <discoveryURL
useType="businessEntity">http://uddi.microsoft.com/discovery?businesskey=8fcc8ad7-e3e4-47b6-85e1-
5c97836e4b75
</discoveryURL>
    </discoveryURLs>

    <name xml:lang="en">DPNM</name>
    <description xml:lang="ko">Research Laboratory in Network Management</description>
```

```
 <contacts>
        <contact useType="">
                <personName>Reasearcher DHKIM</personName>
                <phone useType="">011-524-5772</phone>
                <email useType="">dhkim03@postech.ac.kr</email>
        </contact>
  </contacts>
   <businessServices>
        <businessService serviceKey="a4992886-8f66-443f-98bd-39164a1dec68" businessKey="8fcc8ad7-
e3e4-47b6-        85e1-5c97836e4b75">
          <name xml:lang="en-us">NETCONF_AGENT</name>
        <bindingTemplates>
                <bindingTemplate bindingKey="127f16d0-6681-4e6f-ae91-0eb2f4237623"
serviceKey="a4992886-8f66-                              443f-98bd-39164a1dec68">
                <accessPoint URLType="http">http://141.223.82.6:5454</accessPoint>

                <tModelInstanceDetails>
                        <tModelInstanceInfo tModelKey="uuid:51838e83-8998-4b15-9c4f-
bd5165405e7d">
                                <instanceDetails>
                                        <overviewDoc>

        <overviewURL>http://141.223.82.6:8080/xcms-ws/Netconfd.wsdl</overviewURL>
                                        </overviewDoc>
                                        <instanceParms>UDDI</instanceParms>
                                </instanceDetails>
                        </tModelInstanceInfo>
                        <tModelInstanceInfo tModelKey="uuid:51838e83-8998-4b15-9c4f-
bd5165405e7d">
                                <instanceDetails>
                                        <instanceParms>UDDI</instanceParms>
                                </instanceDetails>
                        </tModelInstanceInfo>
                </tModelInstanceDetails>

                </bindingTemplate>
   </bindingTemplates>
   </businessService>
```

표 12 : DPNM UDDI 레지스트리

businessEntity의 이름이 DPNM으로 DPNM의 이름으로 NETCONF_AGENT 서비스가 등록되어 있음을 볼 수 있다. 각 서비스의 tModelInstanceInfo에는 각 서비스별 인스턴스가 등록되는데 여기의 인스턴스로 각각의 NETCONF 에이젼트를 등록한다. overviewURL에는 실제 에이젼트 서비스의 wsdl이 등록된다.

XCMS-WS가 관리하게 되는 웹 서비스화 되어 등록된 NETCONF 에

이젼트의 UDDI 레지스트리의 구조는 그림 8같다.

XCMS-WS의 NETCONF 매니져에서는 UDDI에 등록된 NETCONF 에이젼트를 찾아서 관리 가능한 장비로 등록한다. 이 과정은 find_bussiness API를 통해 businessEntry XML 데이터를 가져온다. 여기에 포함된 bussinessService의 NETCONF_AGENT를 찾아 그 엔트리의 리스트를 추출한다. 엔트리의 정보는 tModelInstanceInfo에 포함되며 overViewURL에서 가리기는 WSDL문서를 통해 NETCONF 에이젼트 서비스의 명세를 알 수 있다.

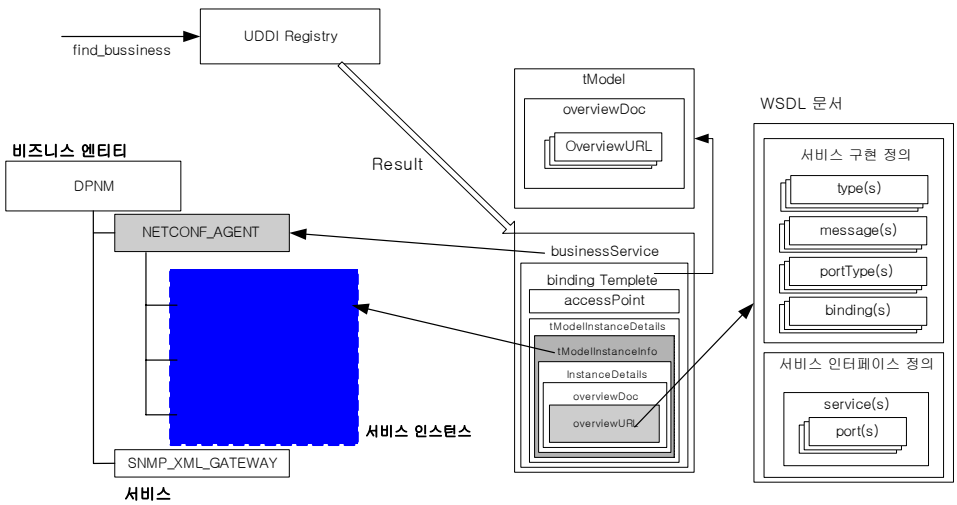


그림 8 : UDDI에 등록된 서비스

우리는 로컬 인트라넷에서 에이젼트 관리가 목적이므로 Private UDDI를 사용한다. UDDI 산업 군 내의 분류가 네트워크 관리에 관한 비즈니스 분류를 갖지 않으며, Public UDDI 에 등록하면 로컬의 장비를 전 인터넷에 공개하는 것이므로 보안 문제가 발생할 수도 있으므로, 이는 허

가된 관리자에 의해 장비를 관리 하게 되는 구성 관리 시스템의 구조에 맞지 않기 때문이다. 또한 Private UDDI 는 UDDI 사양에 정의된 API 이상을 제공할 수 있어서 관리를 위한 특화된 기능 확장이 가능하다.

Private UDDI에는 DPNM의 웹 서비스인 SNMP_XML_GATEWAY라는 SNMP 에이젼트를 XML기반의 매니져에서 사용하기 위한 GATEWAY를 웹 서비스화 한 것이다. 이와 같이 UDDI에는 구성 관리뿐만 아니라 네트워크 매니지먼트를 위한 요소들을 웹 서비스화를 통한 모듈화, 서비스화가 가능하고 이를 본 시스템에서는 이를 이용한다.

3.3. XCMS-WS의 구조

본 논문의 시스템인 XCMS-WS에서 제시하는 구성 관리 시스템의 아키텍쳐는 크게 매니져와 에이젼트, 그리고 UDDI 레지스트리의 3단계로 구성된다. 이번 절에서는 각 단계 별로 이를 구성하는 요소를 살펴본다.

3.3.1. MANAGER

매니저는 크게 5가지로 구성된다. 그 요소는 UI, 메시지 생성 모듈, UDDI Finder, Dynamic Invoker, SOAP 엔진이다. 이 요소를 포함하는 매니져 구조는 그림 9와 같다. UI는 NETCONF 메시지 생성을 위해 관리자에 의해 사용된다. XCMS-WS에서는 NETCONF 에서 필수로 요구되는 CLI와 추가로 웹을 통한 GUI환경이 제공된다. UI를 통해 입력된 정보는 메시지 생성기를 통해 표 9와 같은 NETCONF 프로토콜 메시지로 생성된다. CLI와 GUI에 대한 설명은 4.2와 4.3에서 다루어

진다.

XCMS-WS에서 관리 가능한 에이전트는 UDDI에서 발견하고 사용한다. 이 방법은 UDDI Finder모듈을 통해 이루어지며 세부 방법에 대해서는 3.2.3.에서 언급한바 있다. 모듈에서 에이전트 정보를 가져오고 에이전트의 서비스 명세인 WSDL의 위치를 알 수 있게 되면 WSDL을 통한 원격 호출이 가능하게 된다. 이를 가능하게 하는 것이 Dynamic Invoker이다. WSDL의 명세를 이용해 앞 과정에서 생성된 전달할 NETCONF 메시지를 에이전트로 전달하기 위한 서비스 호출 과정을 Dynamic Invoker가 처리 하게 된다. 또 부가 적인 기능으로 에이전트에게 원격으로 로딩할 구성 정보 파일과 이들을 저장하는 저장 데이터 베이스 모듈 등도 매니져에 포함된다.

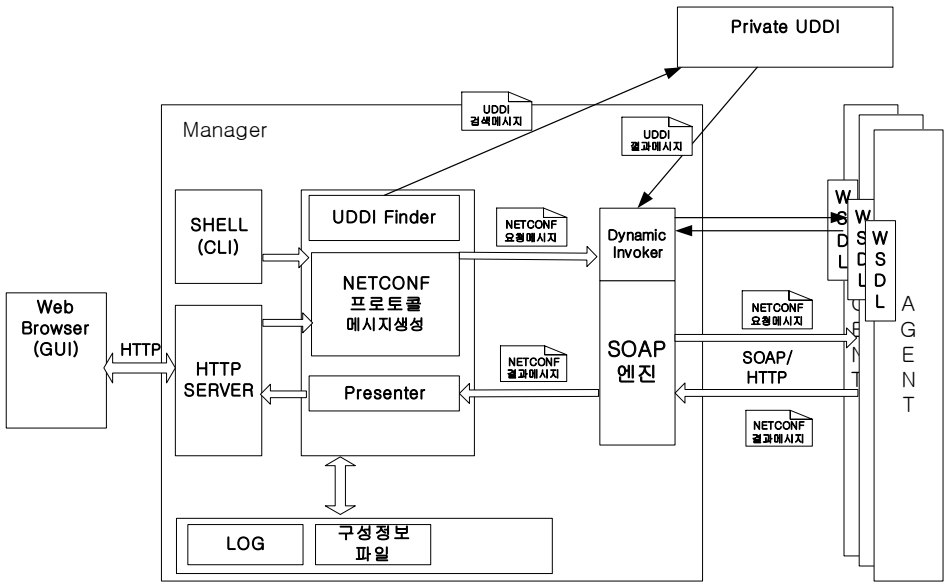


그림 9 : Manager 구조

에이전트는 이 메시지의 분석을 통해 해당 오퍼레이션을 수행하고 결과를 매니져에게 되돌려 주게 된다. 따라서 매니져는 해당 오퍼레이션을 위한 NETCONF 프로토콜 메시지를 생성해야 하는데 그 생성을 위한 메시지 생성기가 XCMS-WS에 포함된다. UDDI Finder를 통해 검색된 에이전트의 WSDL과 메시지 생성기에서 생성된 메시지는 Dynamic Invoker를 통해서 Proxy생성을 통해 에이전트의 서비스를 호출하고 메시지를 전달하게 된다. 이때 에이전트의 처리 과정을 거친 응답 메시지는 Presenter를 통해 GUI로 표시된다.

3.3.2. AGENT

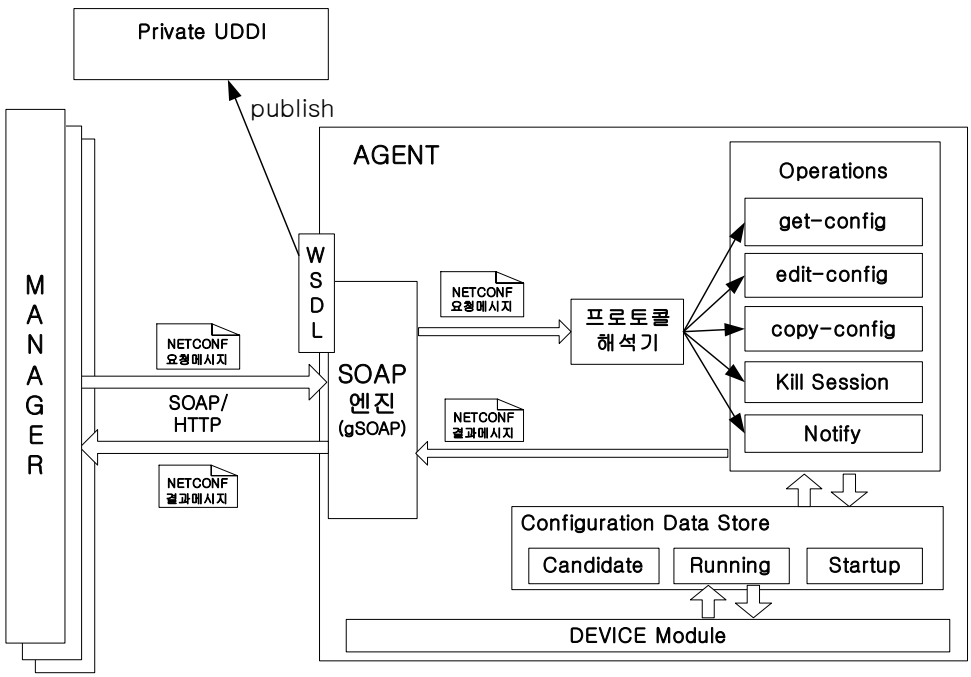


그림 10 : Agent 구조

XCMS-WS 에서의 NETCONF의 관리 프로토콜 메시지는 요청한 서비스를 위해 수행해야 할 오퍼레이션을 바로 호출 할 수 있는 RPC방법을 XML 태그를 사용하여 정의한다. 이 메시지가 매니져로부터 생성되어 에이젼트로 전달 되는데 에이젼트는 이 메시지를 분석하여 구성 관리를 행하게 된다. 따라서 에이젼트에서는 메시지를 전달 받기 위한 통신 모듈인 SOAP엔진과 메시지를 해석하기 위한 프로토콜 해석기, 해석된 메시지의 정보로 실제 오퍼레이션이 호출되며 오퍼레이션은 구성 관리를 수행하기 위한 실제 서비스로서 에이젼트 측에 구현되어 있다. 구성 정보 모델로서 candidate, running, startup에 해당하는 저장소가 있으며 이중 Running은 실제 디바이스 모듈과의 동기화가 되고, 이 정보가 현재 동작 중인 상태를 나타낸다.

SOAP엔진 통한 NETCONF 프로토콜 메시지의 처리 과정에는 에이젼트에 도착한 메시지를 처리 하는 과정의 지연의 문제점 때문에 메시지 큐와 같은 구조가 필요하다. 메시지 큐로 인해 여러 매니져로부터의 요청 및 처리 지연에 따른 메시지 손실의 문제를 해결할 수 있으며 lock과 함께 구성 정보의 동기화의 문제를 해결하기 위한 구조도 제공해 준다. 이는 에이젼트에서 관리하는 연결 정보인 Session번호와 Session별로 관리 되는 메시지 번호가 있음으로서 가능하다.

다음 단계인 NETCONF 프로토콜에 관련된 중요한 모듈인 프로토콜 해석기는 XML 태그로 표현된 오퍼레이션들의 이름과 오퍼레이션 수행에 필요한 파라메터들을 요청 메시지로부터 추출한다. 이 정보들로부터 오퍼레이션을 호출을 하게 된다. 오퍼레이션은 아래와

같다.

<get-config>: 에이전트가 관리하고 있는 구성정보의 특정 부분을 또는 전체를 불러 올 때 사용하는 오퍼레이션이다. 즉 에이젼트가 가지고 있는 구성 정보를 매니저에게 전송하고자 할 때 호출 된다.

<edit-config>: 실제 구성 관리의 실질적인 일을 하는 오퍼레이션이다. 에이전트가 갖고 있는 구성 정보를 변경할 때 수행되는 세 가지 오퍼레이션을 가진다. merge는 새로운 구성 정보를 추가할 때, replace는 기존 구성 정보를 변경 할 때, delete는 기존 구성 정보를 삭제 할 때 호출 된다.

<copy-config>: 매니저가 에이전트에게 구성 정보를 원격으로 로딩하거나 에이젼트의 Running Data Store를 Startup Data Store로 아니면 그 반대의 경우를 수행할 수 있는 오퍼레이션이다. copy-config가 구성관리 정보의 일부분을 조작한다면 이 오퍼레이션은 전체의 백업 및 리스토어가 가능하게 한다.

<delete-config> : Running 이나 Startup, Candidate Data Store의 삭제를 하는 오퍼레이션이다. 물론 이를 위하여 Data Store가 잠김이 되어 있지 않음을 전제로 한다.

<kill-session>: 세션과 관련된 오퍼레이션들 중, 특정 세션을 강제적으로 닫을 때 수행하는 오퍼레이션이다. 세션 아이디는 세션 생성시에 에이전트에서 유일한 값으로 만들어 져서 매니저에게 보내지며, kill-

session 수행시에는 세션 아이디 번호에 해당하는 세션을 멈춘다.

오퍼레이션의 수행 과정에서 각 오퍼레이션은 구성 정보 저장소의 정보를 읽고 변경하는 등의 작업을 하게 된다. 그 수행 결과는 NETCONF 프로토콜 메시지에 담아서 매니져로 보내게 된다.

XCMS-WS에서는 NETCONF 에이젼트에서 관리할 구성 정보를 모듈로서 추가 할 수 있게 하였는데 NG-MON 시스템의 구성 정보도 모듈화된 구조에 의하여 추가되어 있다. 다른 구성 정보도 관리 하기 위한 확장성을 제공한다.

3.3.3. UDDI 저장소

네트워크 매니지먼트에서의 구성 관리는 에이젼트의 구성 정보가 매니져에 의해 자동으로 관리되어야 하는 책임이 있다. 이를 위하여 본 논문에서 제시하는 UDDI 등록 구조는 에이젼트의 구성 정보 이외에 관리 하고 있는 에이젼트의 정보가 변화 될 시 이를 재 등록 과정 등의 관리상에 문제점 없이 매니져 에이젼트간의 구성 정보 관리 서비스 제공해 줄 수 있다. 그림 11에서 보여주 듯 에이젼트의 주소 변경시 UDDI 레지스트리에 대한 정보를 에이젼트 측에서 변경해 놓으면 매니져에서는 다시 등록하는 과정 없이 자동으로 변경점이 반영되는 레이아웃을 제공해 준다.

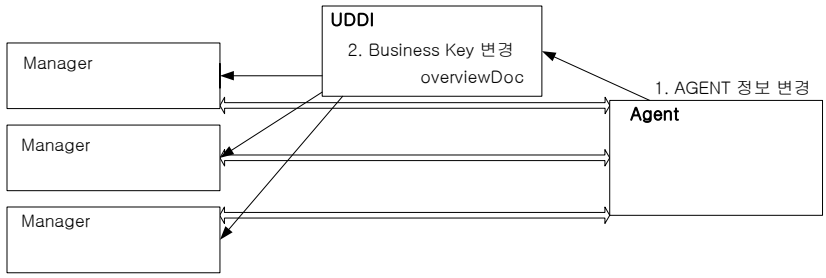


그림 11 : UDDI를 통한 에이전트 변경의 적용

4. XCMS-WS 의 구현

본 논문에서는 NETCONF에서 진행되고 있는 관리 프로토콜을 기반으로 확장된 프로토타입의 웹 서비스 기반의 구성 관리 시스템 구현에 결과를 제시한다. 이 시스템의 유용성을 증명 위해 우리는 3장에서 제시한 구조를 바탕으로 NG-MON 시스템의 구성 정보를 관리하도록 XCMS-WS 시스템을 적용 하였다.

4.1. 구현 환경

매니져

매니저는 에이전트에 비해서 컴퓨팅 자원에 대한 문제가 없고 자바에서 제공되는 다양한 XML관련 API 및 XMLDB[30], 서블릿 컨테이너, SOAP엔진 등 이용하기에 효율적인 Redhat LINUX 9 플랫폼(커널 버전 명시)을 사용하였다. 웹 서비스 및 SOAP엔진으로 Apache[27]그룹에서 제공하고 있는 자바 환경의 AXIS[31]가 작고 성능이 좋으므로 매니저에 들어갈 SOAP/HTTP로 AXIS를 선택하였다. 또 자바 모듈을 웹에서 사용하고 AXIS와 연동할 서블릿 컨테이너로써 역시나 Apache 그룹의 TOMCAT[35]을 사용하였다. 이를 통해 SOAP Over HTTP를 위한 환경이 갖추어 지며 프로토콜 생성 및 서비스 호출에 따른 NETCONF 프로토콜 메시지를 SOAP 프로토콜 메시지의 페이로드에 포함 시켜 보내는 과정이 이루어 진다. 서비스 호출시는 WSDL을 가지고 Dynamic Invocation 환경 구성을 하였다. 이를 위해 WSIF[32]를 이용한다. 이를 통해 에이젼트의 서비스가 명시된 WSDL만 있어도 매니져 측에 서비스의 배치 없이 자동으로 호출 가능하다. 즉

기존의 jws파일의 주소 지정에 의한 웹 서비스 호출과 class의 WSDD [33]를 통한 수동 배치는 서비스의 호출 전 endpoint를 미리 알 경우에만 가능 하지만 우리 시스템은 UDDI에서 검색된 WSDL파일을 통한 NETCONF 에이젼트의 동적 호출이 실제 가능하다. 또한 GUI가 아닌 CLI도 제공하며 이는 gSOAP[39] 및 libxml[41]을 이용해서 자원이 부족한 환경을 위한 Shell에서의 명령어 체계를 갖추었다. 에이젼트에서도 이와 동일한 환경을 통한 시스템 자원 사항이 좋지 않아도 동작이 가능하도록 설계 되었다.

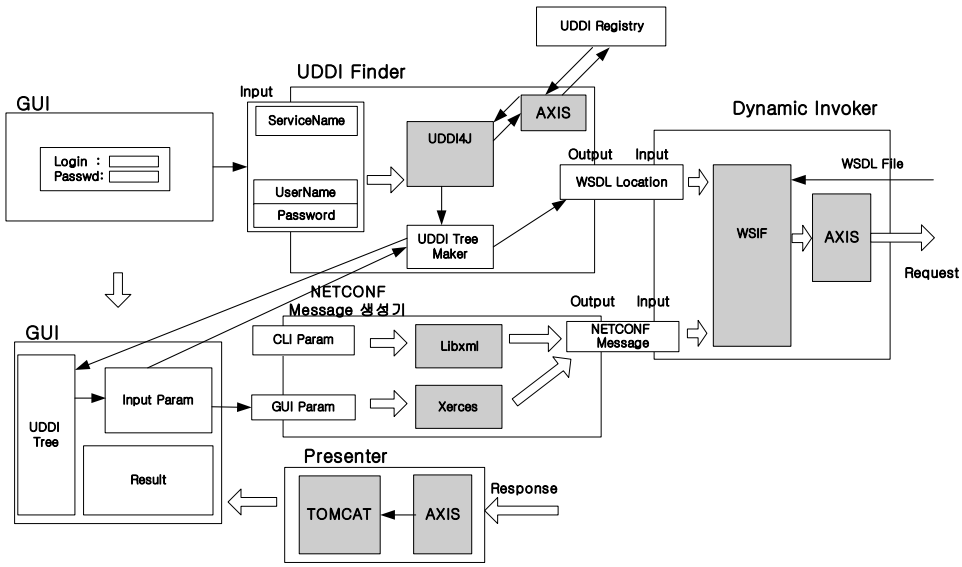


그림 12 : Manager 구성도

에이젼트

에이전트의 경우는 컴퓨팅 자원을 최대한 아껴야 하므로 C 기반의 파서와 SOAP 엔진을 사용하였다. gSOAP은 C/C++환경에서 제공하고

있는 SOAP/HTTP 소스들 중에서 가장 작고 효율적인 SOAP 구현 환경을 제공한다. 또한 다른 SOAP 구현물과 상호 운영성을 보장하고 있다.

gSOAP은 매니저의 AXIS와 상호 운영성을 보장하므로 매니저와 에이전트간의 SOAP메시지 전송이 가능하다. gSOAP은 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.

따라서 이처럼 SOAP 구현 환경에서 제공되는 툴들을 이용하면 시스템 구현 측면에서 효율적이고 편리해진다. 그리고 에이전트와 매니저에 들어 갈 SOAP 서버 모듈은 gSOAP에서 내장되어 Stand-alone 서버를 통하여 RPC 오퍼레이션 서비스인 웹 서비스(Web Services)를 C/C++환경에서 제공 한다.

에이전트에 제공되고 있는 XML 파서는 파서들 중에서 테스트를 거친 가장 작고 효율적인 파서인 libxml을 이용하여 구현하였다.

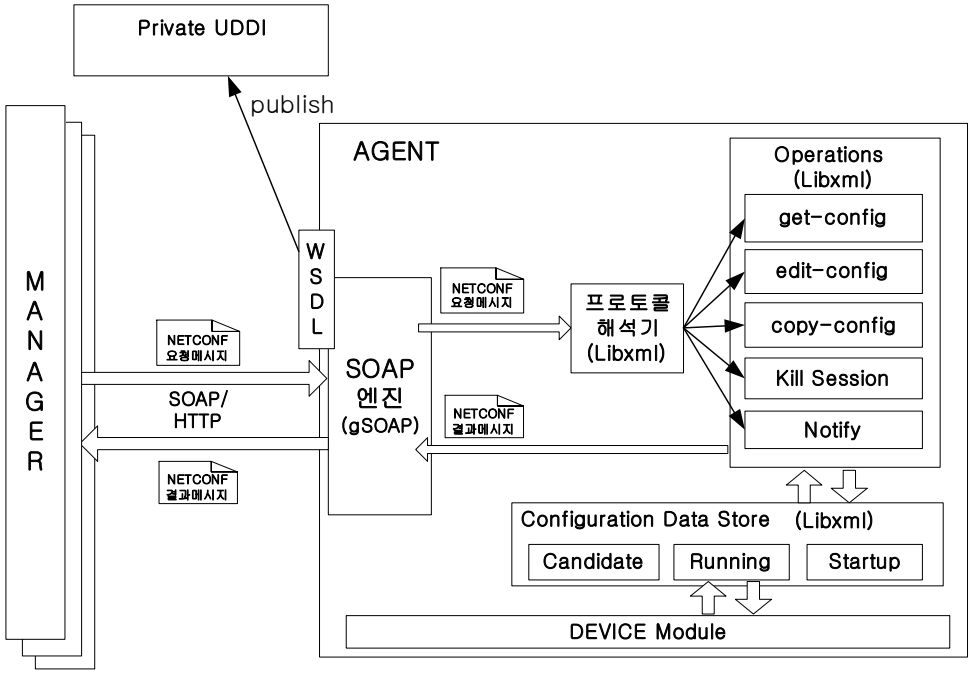


그림 13 : Agent 구성도

UDDI 레지스트리

IBM에서 제공하는 WSDK 패키지[40]에 있는 Private UDDI 레지스트리를 사용한다. Public UDDI에 공표한다는 것은 외부 환경에 내부 장비를 공개하는 것이 되므로 보안상 프라이빗 UDDI Resigtry에 등록하였다.

4.2. CLI(Command Line Interface)

XCMS-WS에서는 Shell에서 사용자의 명령어 입력을 통해 동작 시킬 수 있는 CLI 모드가 존재 한다.

CLI를 통해 앞에서 정의한 NETCONF Operation을 수행하기 위한

메시지 생성이 가능하다. 사용자 명령어를 통하여 실제로 에이전트의 구성 정보와 관련된 RPC 오퍼레이션들이 호출되는 과정을 보여준다. 구성 정보와 관련된 기본 명령어는 다음과 같다.

형식) operationName [hostIP] [database] [xpath] [config]

- operationName : get_config, edit_config
- 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> 수행 시 새롭게 넣어주거나 변경될 구성정보

operationName인 get_config, edit_config등의 Operation이 들어 가며 host IP, database, xpath, config등의 오퍼레이션에 필요한 값이 파라메터로 들어간다. 이 명령어는 메니져에서 분석되어 NETCONF 프로토콜 메시지로 변환된뒤 에이젼트로 보내어진다. 예를들어 'get-config 141.223.82.6 running //admin' 의 명령을 수행하게 되면 아래와 같은 메지지가 생성이되고 141.223.82.6의 NETCONF 에이젼트에 메시지를 전달하게 된다.

```
<get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath">
      //admin
    </filter>
</get-config>
```

GUI 모드에서는 동적 호출을 지원한다. 하지만 CLI모드에서는

WSDL을 통한 동적 호출은 지원하지 않고 에이전트의 직접 호출만이 가능하다. 그 이유로 에이전트의 IP가 파라메터로 필요하게 된다.

4.3. GUI

XCMS-WS는 관리 요소들 중 구성 관리에 특화된 구조를 가지는 시스템으로 WNMS의 구성 관리 부분인 서브 시스템으로 설계 되었으며, WNMS에서는 XCMS-WS 외에도 XML-SNMP Gateway도 웹 서비스 화 되어서 구성 정보 관리 이외의 관리 객체의 접근 및 수정에 쓰이고 있다.

그림 14는 XCMS-WS의 특징인 복잡한 구성 관리가 가능한 확장된 NETCONF 메시지 생성과 UDDI를 이용한 에이전트의 관리를 위해 우리 시스템에서 제시하는 인터페이스를 보여준다.

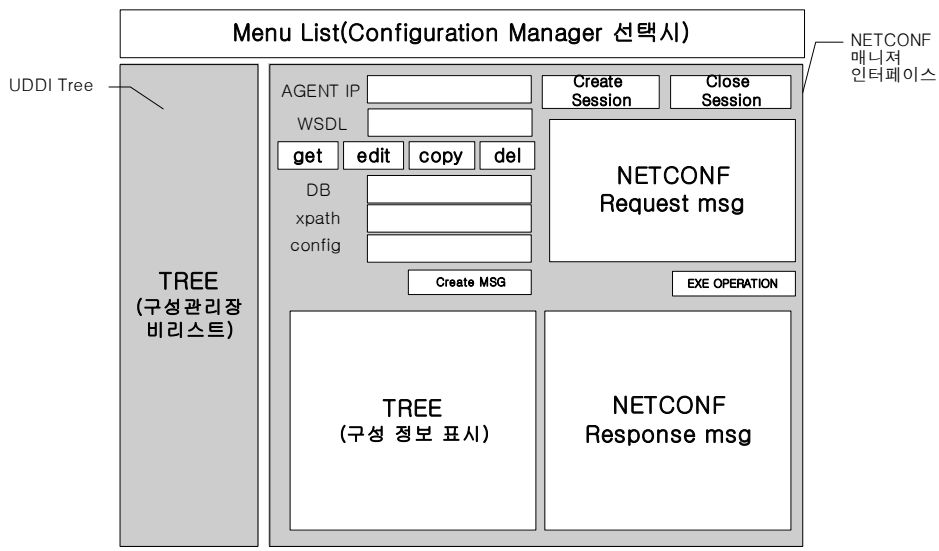


그림 14 : GUI

상단의 Menu List는 WNMS시스템의 여러 서비스를 선택할 수 있는

메뉴이다. 메뉴중에서 'NetConfManager'를 선택하게 되면 XCMS-WS의 구성 관리를 위한 NETCONF 매니져 인터페이스가 나타난다. TREE는 UDDI에 등록된 서비스를 표시하는 창이다. 오른쪽의 창은 NETCONF 프로토콜 명령을 생성하는 관리 창이다. get-config, edit-config, copy-config, delete-config명령을 선택하고 Operation에 필요한 DB, XPath, Config 항목을 입력 받아 'Create MSG'의 버튼을 통해 NETCONF 프로토콜 요청 메시지를 생성할 수 있다. 메시지를 'EXE Opeation'을 통해 선택된 에이전트에 전달한다. 그 결과 메시지는 아래의 NETCONF Response msg 창에 표시되며 TREE에는 구성 정보의 전체 구조가 표시된다.

4.4. 작동 과정의 예

우리는 XCMS-WS시스템의 동작 과정을 보이기 위해 네트워크 트래픽 모니터링 시스템인 NG-MON[42]의 설정 파일을 관리하는 예를 든다.

[작동환경]

XCMS-WS는 매니져, 에이젼트 구조이며 에이젼트 검색을 위한 UDDI 레지스트리가 둘 사이에 위치한다. 또한 에이젼트에서는 NG-MON의 설정 파일을 관리 가능한 구성 정보로 로딩한다.

그림 15에서 보이듯 에이전트는 NG-MON 모듈을 관리 한다. 메니져는 동일한 인터페이스를 통해 에이전트에 메시지를 전달하고 에이젼트는 이를 NG-MON 모듈에 전달하게 된다. 여기서 말하는 NG-MON 모듈

은 NG-MON 시스템의 구성 정보를 관리 하기 위해 NG-MON의 구성 정보파일을 NETCONF 에이젼트에 로딩하고 동기화 시키며, NETCONF 메시지에 따라서 구성정보의 변경등의 처리하는 역할을 한다. 이 구조는 NG-MON이 아닌 다른 구성 정보를 관리 할 때에도 마찬가지로 적용된다.

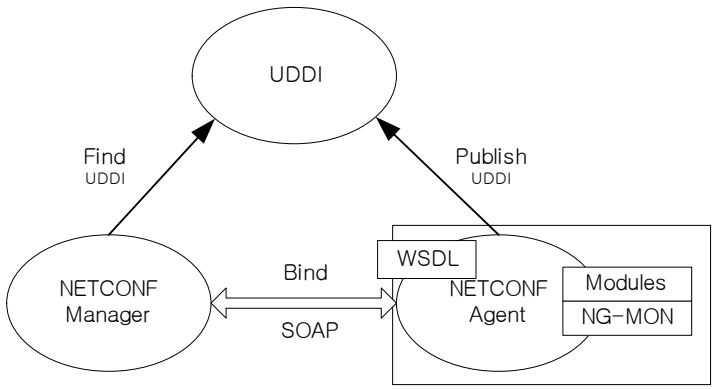


그림 15 : 관리 구조의 예

표 13은 데모에서 관리 하게 될 NG-MON시스템의 구성 정보 XML구조이다. 이 구성 정보를 관리하는 모듈이 에이젼트에 등록되면 매니져 측에서 UI를 통해 관리가 가능하다. 이 구성정보는 에이젼트 동작시 Running구성 데이터 저장소와 동기화 된다.

```
<?xml version="1.0"?>
<ng-mon ip="141.223.82.144">
        <admin name="DongHyun Kim" email="dhkim03@postech.ac.kr"></admin>
        <database user="root" password="********"/>
        <packetcapture>
                <device name="eth1"/>
        </packetcapture>
        <flowgenerator interval="2">

        </flowgenerator>
        <flowstore>
```

```
                    <p2p file="p2p.xml"/>
        </flowstore>
        <analyzer>

        </analyzer>
        <presenter>

        </presenter>
</ng-mon>
```

표 13 : 관리 할 NG-MON 시스템의 구성 정보 파일

[UDDI – 관리 에이전트 탐색 과정]

왼편 TREE에서는 관리자 계정으로 UDDI에 접속하여 UDDI에 등록된 NETCONF 에이전트 및 서비스를 검색하여 표시한다. Business Entity에 저장된 우리 웹 서비스의 정보 중 NETCONF 프로토콜을 이용하는 서비스, 즉 서비스가 동작 중인 Device의 리스트를 가져 온다. 계정을 통해 관리되는 사설 UDDI 레지트리에 등록된 정보를 우리 시스템에서 가져오는 과정에서 필요한 정보를 필터링함으로써 가능하며 이 구조는 UDDI의 내용이 바뀌면 바뀌어진 내용이 우리 시스템에 자동으로 반영되는 동기화된 구조를 보여준다.

표 14은 UDDI검색 결과를 재구성하여 UDDI의 트리에서 쓰이는 XML 파일이다. 표 12의 UDDI 레지스트리 BussinessEntry의 정보에서 Service와 TModelInstance에 포함된 에이전트 이름과 서비스 명세인 WSDL을 추출하여 구성한 것이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessName bizID="DPNM">
  <Service serviceID="NETCONF_AGENT">
    <TModelInst devID="NETCONF_AGENT1" overViewDoc="http://141.223.82.6:8080/xcms-
ws/Netconfd.wsdl"/>
    <TModelInst devID="NETCONF_AGENT2" overViewDoc="http://141.223.82.7:8080/xcms-
ws/Netconfd.wsdl"/>
  </Service>
  <Service serviceID="SNMP_XML_GATEWAY">
    <TModelInst devID="SNMP_XML_GATEWAY"
```

```
overViewDoc="http://141.223.82.6:8080/axis/services/SNMP_XML_GATEWAY?wsdl"/>
  </Service>
</BusinessName>
```

표 14 : UDDI 검색 결과의 재구성

[매니져 – Request과정]

매니져 측에서는 CLI를 통한 입력과 GUI를 이용한 사용자로부터 입력된 파라메터를 가지고 메시지 생성기에서 NETCONF 프로토콜 메시지를 생성한다. 이를 GUI에서는 AXIS, CLI에서는 gSOAP를 이용하여 SOAP메시지에 페이로드로 실어서 에이젼트에 보내게 된다. 이때 매니저 측에서는 Dynamic Invoke 과정이 일어나는데 이는 SOAP엔진이 UDDI Finder에서 발견된 에이젼트의 서비스를 명세한 WSDL파일을 통해 서비스의 원격 Proxy를 생성하게 된다. SOAP 메시지를 에이젼트로 보냄으로써 에이젼트의 웹 서비스를 호출 하게 된다.

[에이젼트 – Response과정]

에이젼트 측에는 NETCONF 프로토콜 처리기가 대기하고 있으며 매니져측에서 전달된 SOAP메시지로부터 NETCONF 메시지를 분리해 내고 NETCONF 프로토콜 해석기는 프로토콜을 해석하여 매니져가 요청한 오퍼레이션 처리 함수로 메시지를 넘긴다. 오퍼레이션에 해당하는 처리를 거친다. 예를 들어 그림 17과 같이 오퍼레이션이 get-config이고 source 구성 정보 저장소 running이면 메모리에 로딩된 현재 상태를 나타내는 running DB에 저장된 XML 스트림을 DOM Tree로 생성하고

XPath에 해당하는 Tree를 역 직렬화를 통해 꺼내어 NETCONF 메시지를 생성하고 SOAP 엔진을 통해 매니져로 전달 된다.

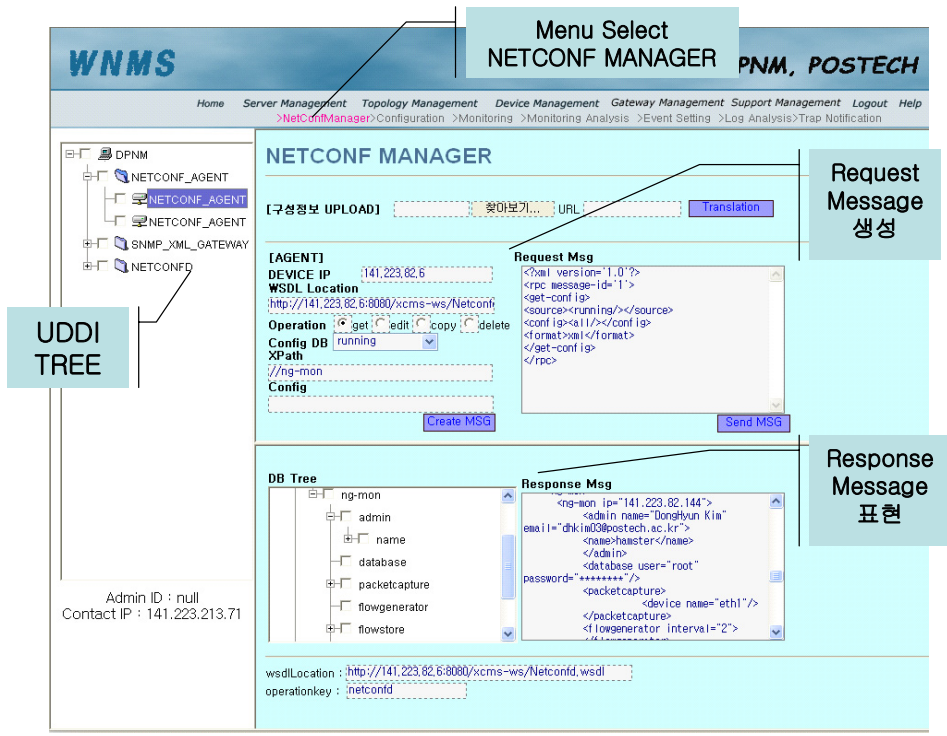


그림 16 : 실제 동작 화면

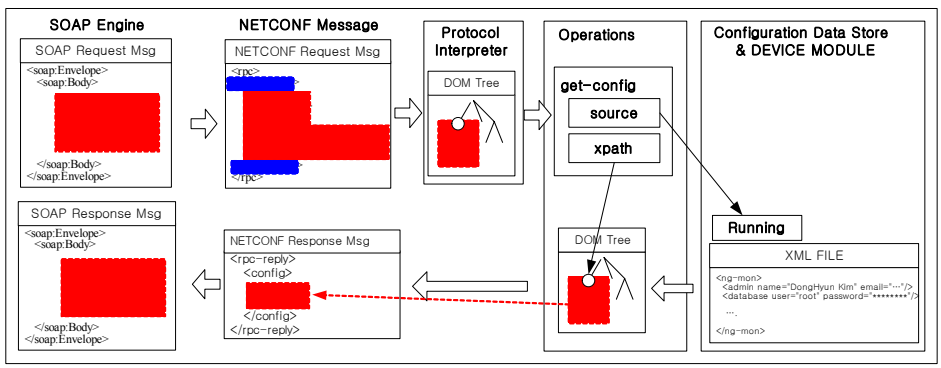


그림 17 : 에이젼트에서 get-config 동작과정

5. 결론 및 향후 과제

구성 관리의 문제점을 해결하기 위한 방법의 하나로 XML기반의 기술들이 논의되었다. 본 논문에서는 그 대표적인 활동인 NETCONF 프로토콜을 다루었다. NETCONF 관련 인터넷 초안 및 그 관련된 시스템을 분석한 결과 우리는 3가지의 문제점을 발견했다. 첫째로 NETCONF 프로토콜에서 구성 정보 수정을 위한 메시지 구조가 addressing에 있어서 비효율적이며 복잡한 구성 정보 수정 시 여러 번의 메시지 전송이 필요하다는 점, 둘째로 SOAP의 유용성에 대해 설명하고 있으나 SOAP Binding에 대한 실제 레퍼런스 구현이 존재하지 않는다는 점, 마지막으로 WSDL, SOAP등의 웹 서비스 기술을 사용하나 이는 메시지 전송만이 고려 되어져 있어서 구성 장비 에이젼트의 발견과 호출에 대한 방법이 제시되어 있는 않는 점이 문제이다.

우리는 기존의 XML 관련 툴 및 웹 서비스 [9]관련 기술들을 충분히 활용하고 앞에 제시된 3가지 문제점을 해결할 수 있는 개선책이 포함된 XCMS-WS시스템을 제시하였다. 이 시스템은 XPath를 사용한 구성 정보의 효율적인 조작 및 NETCONF 프로토콜을 이용하여 구성 정보의 효과적 통신이 가능하며 Private UDDI를 이용한 에이젼트관리 또한 효율적이다.

이번 연구가 프로토콜의 효율적 동작에 초점을 맞춘 구현 이라면 다음 연구는 활용 방안, 즉 다양한 관리 모델에 적용을 하여 보는 것이다.

상용 장비의 tModel등록을 통한 장비별 NETCONF 에이젼트 기본 구성정보 업데이트 자동화 방안이 그 예이다. 또한 프로토콜 자체의 성능 테스트 및 최적화, 임베디드 환경을 위한 최적화 및 포팅, 정보 모델의

표준화를 위한 연구도 향후 과제이다.

References

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

[2] **The Internet Engineering Task Force(IETF), http://www.ietf.org/**.

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

[4] Enns, R., “NETCONF Configuration Protocol”, draft-ietf-Netconf-prot-01 (work in progress), Oct. 2003,< http://www.ietf.org/internet-drafts/draft-ietf-Netconf-prot-01.txt>.

[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] Enns, R., “NETCONF Configuration Protocol” draft-ietf-netconf-prot-04, October 24, 2004,< http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-04.txt>.

[8] T. Goddard, ” Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)”draft-ietf-netconf-soap-03” September 7, 2004 <http://www.ietf.org/internet-drafts/draft-ietf-netconf-soap-03.txt>.

[9] “W3C: Web Services Activity,” <url:http://www.w3.org/2002/ws/>.

[10] OASIS, “Universal Description, Discovery and Integration (UDDI)”, http://www.uddi.org/

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

[12] P. Shafer and R. Enns, JUNOScript: An XML-based Network Management API, http://www.ietf.org/internet-drafts/draft-shafer-js-xml-api-00.txt, Aug. 27, 2002.

[13] Cisco Systems, Cisco Configuration Registrar, http://www.cisco.com /univercd/cc/td/doc/product/rtrmgmt/ie2100/cnfg_reg/index.htm .

[14] GOTTSCHALK ET AL, "Towards Introduction to Web services architecture," IBM SYSTEMS JOURNAL, VOL 41, NO 2, 2002.

[15] T.Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, August 1998.

[16] W3C, "Web Services Description Language (WSDL) Version 1.2" W3C Working Draft, July 2002.

[17] Jonathan B. Postel "SIMPLE MAIL TRANSFER PROTOCOLT" RFC 821 August 1982, http://www.faqs.org/rfcs/rfc821.html.

[18] K. Moore "MIME (Multipurpose Internet Mail Extensions)" RFC2047 November 1996, http://www.faqs.org/rfcs/rfc2047.html.

[19] J. Postel , J. Reynolds "FILE TRANSFER PROTOCOL (FTP)" RFC 765 October 1985, http://www.faqs.org/rfcs/rfc959.html.

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

[21] 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>.

[22] Lear, E., Crozier, K., Enns, R., "BEEP Application Protocol Mapping for NETCONF", draft-lear-Netconfbeep-00 (work in progress), Oct. 2003, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-beep-00.txt>.

[23] Goddard, T., "NETCONF over SOAP", draft-ietf-Netconf-soap-00 (work in progress), Oct. 2003, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-soap-00.txt>.

[24] YENCA, a Netconf agent for Linux implemented in C, http://madynes.loria.fr/Software_gb.htm

[25] Netconf Data Model (netmod) Bof , http://www.ietf.org/proceedings/04aug/184.html

[26] H. M. Choi, M. J. Choi, and J. W. Hong, "Design and Implementation of XML-based Configuration Management System for Distributed Systems", Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, Apr. 2004.

[27] Apache Group, "Apache", http://www.apache.org/.

[28] Apache XML project, "Xerces Java parser", http://xml.apache.org/xerces-j/.

[29] Apache XML project, "Xalan Java", http://xml.apache.org/xalan-j/.

[30] Apache XML project, "Xindice", http://xml.apache.org/xindice/.

[31] Apache XML project, "Axis", http://xml.apache.org/axis/.

[32] Apache WebService project, WebService Invocation Frameworks "WSIF", http://ws.apache.org/wsif/

[33] Webservice Deployment Despricriptor(WSDD) Reference, http://www.osmoticweb.com/axis-wsdd/

[34] Apache XML project, Web Services Invocation Framework "WSIF", http://ws.apache.org/wsif/

[35] Apache jakarta project, TOMCAT http://jakarta.apache.org/tomcat/

[36] W3C, "Document Object Model (DOM) Level 2 Core Specification", W3C Recommendation, Nov. 2000.

[37] Introduce to CORBA, http://java.sun.com/developer/onlineTraining/corba/corba.html

[38] Sun Java System Message Queue, http://wwws.sun.com/software/products-/message_queue/

[39] 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.htm/.

[40] Web services Develop Kit(WSDK), http://www-106.ibm.com/developerworks /webservices/wsdk/

[41] libxml, “The XML C parser and toolkit at Gnome”, http://www.xmlsoft.org/

[42] Se-Hee Han, Myung-Sup Kim, Hong-Teak Ju and James W.Hong, “The Architecutre of NG-MON:A Passive Network Monitoring System”, Lecture Notes in Computer Science 2506, Edited by M. Feridun, P.Kropf and G.Babin, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM2002), Montreal, Canada, October, 2002, pp. 16-27.

[43] OASIS, “Universal Description, Discovery and Integration (UDDI)”, http://www.uddi.org/.

[44] XML:DB, “XML:DB”, http://www.xmldb.org/.

요 약

현재 인터넷의 빠른 확산으로 다양한 네트워크 장치들이 출현하게 되었고, 거대한 네트워크가 형성되었다. 현재 네트워크 관리를 위해 사용되고 있는 SNMP(Simple Network Management Protocol) 기반의 관리시스템은 나날이 복잡해지고 거대해진 네트워크를 수용하기에는 관리 정보 모델과 관리기능 측면에서 취약함이 지적되고 있다. 또한 UDP를 기반으로 한정된 크기의 SNMP 메시지를 보내기 때문에 벌크 데이터(bulk data)를 처리 하는데 있어서 가장 큰 문제가 되고 있으며, 이는 구성관리(Configuration Management)의 어려움을 더욱 가중시키게 된다. 따라서 SNMP의 대체 방법인 XML(Extensible Markup Language) 기반의 네트워크 관리는 구성관리를 중심으로 IETF의 워킹그룹(WG)인 Netconf(Network Configuration)에서 표준화 작업이 활발히 진행되고 있다. 우리는 본 논문에서 Netconf에서 제시한 관리 프로토콜의 문제점을 지적하고 이를 개선한 XML 기반의 구성 정보 관리 시스템(XCMS)의 구조를 제안하였다. XCMS는 관리 프로토콜로써 RPC(Remote Procedure Call) interface를 제공하고, WSDL(Web Services Description Language)을 정의 할 수 있는 SOAP을 이용하고 있다. 그리고 본 논문은 임베디드 시스템인 XCMS의 에이전트 고려하여 관리 정보를 추출하기 위해 필요한 XPath 표현들을 제안하였고, 구성 정보 객체간의 연관성을 고려한 XML기반의 구성 정보 모델링 방법을 제시하였다. 또한 성능 평가를 통하여 기존의 SNMP 방법보다 XCMS가 구성관리 측면에서 효율성을 증명하였다. 우리는 현재 XCMS 의 성능 및 효율성 향상을 위하여 XCMS 에이전트를 최적화하는 작업을 하고 있다. 그리고 더 나아가 새로운 분산 환경을 제공하고 있는 Web Services 기술을 XCMS에도 적용해 보는 연구 또한 진행되고 있다.

감사의 글

석사 2년 동안 부족한 저를 지도해 주시고 돌봐주신 홍원기 교수님께 진심으로 고개 숙여 감사 드립니다. 교수님의 가르치심은 언제나 저에게는 삶의 밑거름이 될 것입니다. 더불어 저의 논문 심사를 위해 애써 주신 김종 교수님과 서영주 교수님께도 깊은 감사를 드립니다.

짧은 석사기간 동안이었지만 연구실 선후배 및 동료 여러분 그동안 고마웠습니다. 연구실 생활에 많은 도움을 주고 제 연구에 관심을 갖고 돌봐준 명섭선배 감사합니다. 좋은 논문 쓰시고 곧 졸업하시길 바라겠습니다. 매번 영어 논문 쓸 때 마다 함께 고생하고 제게 힘이 되어준 미정언니 너무나 감사합니다. 외국에서의 포스닥 생활도 지금처럼 즐겁고 힘차게 하시길 바라겠습니다. 포항의 후배들을 위해 끊임 없는 관심을 보여주시는 홍택 선배님과 석사 1년차 때 동기처럼 같이 어울렸던 효진이, 윤정언니, 훈정선배, 세희선배에게도 감사의 마음을 전합니다. 그리고 항상 따뜻하게 옆에서 응원해 준 착한 후배 소정이와 뒤늦게 우리 팀에 합류하여 많은 도움을 준 동현선배, 그리고 즐겁게 연구실 생활을 할 수 있게 도와준 후배 성철, 승화, 영준, 형조에게도 고마움과 함께 앞으로 남은 석사 생활 잘 하길 바랍니다. 2% 부족한 우리의 연구실 생활을 채워주는 은희 언니와 열심히 자신의 일을 하는 멋진 영미 언니에게도 감사합니다. Nice to meet you. Thank you, Deepali. I hope you have a good time in DPNM. 헤어지게 되어 아쉽지만, DPNM 연구실에서의 보낸 시간들은 제게 너무나도 소중하고 감사합니다.

작은 고민이라도 마음 터놓고 생활했던 여자 기숙사 2동 209호 여전사들 문경, 주영, 그리고 평생 내 옆에서 든든한 벗이 되어줄 수영, 미섭, 영란에게도 감사의 마음을 전합니다.

지금까지 뒷바리지 해준 부모님과 물심양면으로 힘이 되어준 큰오빠, 큰언니, 작은 오빠에게 그리고 곧 세상에 태어날 첫 조카에게도 감사하다는 글을 올립니다.

이 력 서

성 명 : 김 동 현

생 년 월 일 : 1975년 5월 30일

출 생 지 : 경상남도 김해시

주 소 : 부산시 수영구 광안1동 상아아파트 103동 1035호

학 력

1995.3 – 2002.2 : 경성 대학교 컴퓨터 공학과 (B.S.)

2002.3 – 2005.2 : 포항공과대학교 컴퓨터 공학과 (M.S.)

학 술 활 동

♦Conference Papers

- Hyoun-Mi Choi, Mi-Jung Choi, James W. Hong, "Design and Implementation of XML-based Configuration Management System for Distributed Systems", Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 2004.
- 최현미, 최미정, 홍원기, 박용석, “NETCONF 표준을 따른 XML 기반의 네트워크 구성관리 시스템”, Accepted to appear in the KNOM Review, Vol.6, No.2, December 2003.
- Hyoun-Mi Choi, Mi-Jung Choi James W. Hong, "XML-Based Configuration Management for Distributed System", Proc. of 2003 Asia-Pacific Network Operations and Management Symposium (APNOMS 2003), Fukuoka, Japan, October 1-3, 2003, pp. 599-600.
- 최현미, 최미정, 홍원기, "분산 시스템의 구성관리를 위한 XML 기반의 관리 시스템 설계 및 구현”, Proc. of KNOM 2003 Conference, Daejeon, Korea, May 22-23, 2003, pp. 330-337.

연구 활동

♦Projects

- Development of XML-based Network Management System (XNMS) (XML 기반의 네트워크 관리 시스템 개발), DPNM Project, 2003
- Development of XML-based Configuration Management System for IP Network Devices (XCMS) (네트워크 장비를 위한 XML 기반의 구성 관리 시스템 개발), DPNM Project, 2003