웹 서비스를 활용한 NETCONF 구성관리 시스템

[1]유선미, [2]주홍택, [1]홍원기

[1]포항공과대학교 컴퓨터공학과

[2]계명대학교 컴퓨터공학과

{sunny81, jwkhong}@postech.ac.kr, juht@kmu.ac.kr

요 약

인터넷이 발전함에 따라 네트워크의 규모는 나날이 증가하고 있으며, 인터넷을 구성하는 IP 기반의 네트워크는 다양한 회사의 제품들로 구성되어 있다. 이러한 환경 속에서 대다수의 네트워크 장치를 관리하기 위한 구성관리(Configuration Management)에 있어서 취약점이 지적 되어왔다. 전통적으로 네트워크 장치에 대한 구성관리(Configuration Management)는 회사마다 다른 방식의 문자기반의 CLI (Command Line Interface)가 주로 사용되어, 여러 회사의 제품들로 이루어진 네트워크에 대한 구성관리가 비효율적으로 이루어 지고 있었다. 국제 표준화 단체인 IETF 에서는 여러 회사 제품들간의 상호 운용성을 확보하기 위하여 네트워크 장치 제조사에 의존적이지 않은 XML 기반의 구성관리 표준을 NETCONF 작업반(Network Configuration Working Group)에서 제정하고 있다. 이 작업반의 작업결과로 NETCONF 인터넷 초안 (Internet Draft)이 발표되고 있다. 본 논문은 NETCONF 에서 진행 중인 인터넷 초안을 분석하여 개선할 점을 제시한다. 우리가 제시하는 NETCONF 의 개선점은 XPath 를 사용한 구성 정보의 효율적인 조작방법이다. 또한, Private UDDI 를 이용하여 새로운 네트워크 장비가 추가되거나 각 장치의 구성관리 소프트웨어가 변경되었을 경우에 쉽게 적응할 수 있는 방법도 제시한다. 본 논문에서 NETCONF 인터넷 초안을 기반으로 개발한 구성관리 시스템의 개발결과도 제시한다. 제시하는 NETCONF 기반의 구성관리 시스템 개발 결과는 아직 표준화 단계에 머물러 있는 NETCONF 프로토콜의 실제 구현 사례로 사용될 수 있다.

1. 서론

인터넷은 다양한 회사의 네트워크 장치들 간의 연결로 구성되어 있다. 이러한 환경 속에서 대다수의 네트워크 장치를 관리하기 위한 구성 관리(Configuration Management)는 운용자에게 의존적인 CLI(Command Line Interface)가 주로 사용됨으로써 비효율적인 방법으로 이루어져 왔다. 이 구성 관리의 문제점을 해결하기 위한 방법의 하나로 XML(Extensible Markup Language)[1] 기반의 기술들이 논의되었다. 그 유용성이 표준화 기관들과 벤더들을 통해 논의되고 증명되었으나 이 과정에서 상호 운용성을 확보하기 위하여 표준화에 대한 필요성이 제기되었다. IETF(Internet Engineering Task Force)[2]는 NETCONF 작업반(Network Configuration Working Group)[3]을 구성하여 이에 대한 표준화 작업을 진행하고 있다. NETCONF 는 관리 오퍼레이션들과 매니저와 에이전트간에 전송되는 메시지 포맷인 관리 프로토콜에 대하여 표준화하고 있다. 이러한 관리 프로토콜의 표준화 작업은 여러 업체에서 공급되는 다양한 네트워크 장비들의 에이전트들과 매니저간의 관리 오퍼레이션을 위한 상호운영성을 보장해 주게 된다.

NETCONF 는 네트워크 관리에 있어서 크게 두 가지 부분에서 사용되는데 하나는 정보 모델 즉, 복잡한 관리 정보의 모델링하는 것이고, 또 다른 하나는 매니저와 에이전트간의 통신 모델 즉, 메시지 포맷인 관리 프로토콜을 정의하는 것이다.

NETCONF 의 관리 프로토콜 메시지는 요청한 서비스를 위해 수행해야 할 오퍼레이션을

바로 호출 할 수 있는 RPC(Remote Procedure Call)방법을 XML 태그를 사용하여 정의한다. 즉 정의 된 오퍼레이션들의 이름은 XML 태그로 표현되어 있어 에이전트측에서는 이 메시지를 분석하여 오퍼레이션을 실행하고 결과를 XML 형식의 메시지에 담아 보내게 된다. 이와 같은 방법을 통해 XML 메시지로 RPC 호출이 가능하게 된다. NETCONF 에서는 RPC 를 제공하기 위한 방법 중 하나로 W3C(World Wide Web Consortium)에서 정의한 SOAP(Simple Object Access Protocol)[5]메시지를 이용한다. SOAP 에서, 관리 프로토콜의 메시지는 전송 프로토콜의 데이터 (payload) 부분에 들어가기 때문에 전송 프로토콜에 대해 의존적일 필요가 없다. 그러나 전송 프로토콜로 HTTP[4]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.

NETCONF 에서는 업데이트 된 새로운 인터넷 초안(Internet Draft)[6],[7]을 발표하였으며, 여기에는 이전의 NETCONF 프로토콜의 개선점 및 NETCONF 전송 계층의 하나인 NETCONF over SOAP[7]에 대한 정의가 포함되어있다. NETCONF 에 대한 연구로서 오픈 소스 진영에서는 프랑스 Loria 연구소의 YENCA 프로젝트[18]가 진행되고 있다. 하지만 이는 NETCONF 프로토콜에 대한 구현일 뿐이며 실제 NETCONF over SOAP[7]에 대한 고려는 되어 있지 않다.

NETCONF 관련 인터넷 초안 및 그 관련된 시스템을 분석한 결과 우리는 두 가지의 문제점을 발견했다. 첫째로 NETCONF 프로토콜에서 구성 정보 수정을 위한 메시지 구조가 addressing 에 있어서 비효율적이며 복잡한 구성 정보 수정 시 여러 번의 메시지 전송이 필요하다는 점이다. 둘째로 WSDL, SOAP 등의 웹 서비스 기술을 사용하나 이는 메시지 전송만이 고려 되어 있을 뿐 구성 장비 에이전트의 발견과 호출에 대한 방법이 제시되어 있지 않다. 메시지 구조의 복잡성은 XPath[10] 기술을 활용하여 해결할 것이며 구성관리를 위한 장치의 발견 및 호출은 UDDI[9] 기술을 활용하여 해결한다.

NETCONF 표준에 기반하여 우리가 개발한 XCMS-WS(XML-Based Configuration Management System using Web Services) 시스템은 최근의 NETCONF 표준을 모두 준수하고 있으며 앞에서 제시한 개선방안도 포함되어 있다. XCMS-WS 는 아직 표준화 단계에 머물러 있는 NETCONF 표준에 대하여 실제 적용할 수 있는 구현 사례이다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로써 웹 서비스 기술을 소개하고, 구성 관리 시스템과 그 최근 연구들을 알아 본다. 또 현재 표준화 진행 중인 NETCONF 프로토콜에 대해 고찰한다. 3 장에서는 NETCONF 프로토콜의 문제점을 해결하기 위한 설계 구조 및 시스템 설계에 필요한 요구사항들을 제시하며 4 장에서는 시스템 구현 환경, 동작 과정을 보여 주며, 시나리오로 GUI(Graphic User Interface)를 통한 시스템 동작의 예를 보인다. 마지막으로 5 장에서는 결론과 향후 연구에 대해 기술한다.

2. 관련 연구

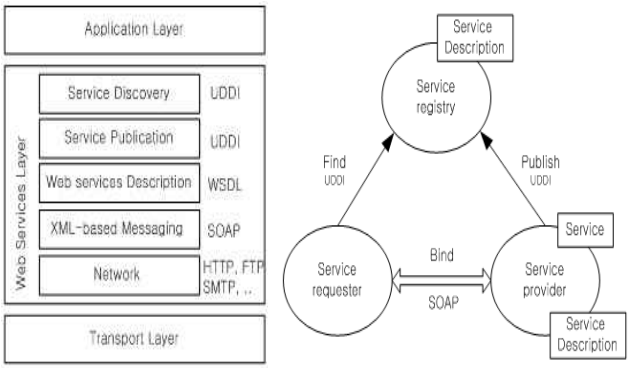
이 장에서는 본 연구의 결과로 구현된 XCMS-WS 시스템에 적용되는 관련 연구로 웹 서비스 기술과 웹 서비스 기반의 네트워크 관리 시스템을 소개한다. 구성 관리의 선행 연구로서 Cisco 나 Juniper Networks 같은 장비 업체에서 개발된 XML 기반 구성 관리 시스템[11],[12]에 대해서 알아 보고 이의 문제점을 해결하기 위한 IETF 에서 표준화 진행 중인 NETCONF 프로토콜을 소개한다.

2.1. 웹 서비스 기술

웹 서비스 아키텍처는 인터넷을 통한 애플리케이션간의 통신을 위해 W3C 에 의해서 정의되었다. 웹 서비스는 인터페이스와 바인딩을 URI(Uniform Resource Identifiers)[13]로 정의하고 XML 을 이용하여 구현한다. 웹 서비스는 WSDL(Web Services Description Language)[14]을 통해 외부에 공개 되고 발견될 수 있으며 인터넷 프로토콜 위에 XML 메시지로서 상호 작용이 가능하도록 한다. 플랫폼, 언어에 관련 없는 표준 웹 기술에 의한 컴포넌트간의 접근이 가능하다.

웹 서비스는 전송 계층과 애플리케이션 계층 사이에 위치한다. 웹 서비스 레이어 자체는 그림 1(a)에 표현 한 것과 같이 WSDL, SOAP, UDDI, HTTP 등 여러 인터넷 표준 프로토콜에 기초한다. 이중 상호작용을 위한 첫 번째 레이어는 HTTP[4]로 메시지를 전송하는 역할을 한다. 두 번째 레이어는 SOAP 으로 HTTP 와 함께 HTTP Post 방식에 의한 페이로드의 전송으

로 RPC 호출을 한다. 이때 세 번째 레이어에서 쓰이는 WSDL[14] 문서는 서비스의 명세인 웹 서비스의 공용 인터페이스 정의를 제공한다. 마지막으로 UDDI[9]는 이 공용 인터페이스 정의를 공개적으로 등록 가능하게 하며 이의 검색을 통해 공개된 서비스의 접근을 위한 방법을 제공한다.



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

그림 1: 웹 서비스 구조

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

2.2. 구성 관리 시스템

XML 을 이용하여 네트워크 관리를 하는 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. 특히 XML 을 기반으로 한 RPC 는 장비들과의 데이터 교환을 위한 간단하고 호환성 있는 기술을 제공한다.

Juniper 의 JUNOscript[11]와 CISCO 의 Configuration Registrar[12]등이 이러한 방법론의 예이다. Juniper Networks 의 JUNOScript 는 JUNOS NOS(Network Operating System)에서 시스템의 조작을 위한 스크립트로써 처음 소개 되었으며, XML 을 기반으로 한 네트워크 장비의 관리 노력과 비용을 줄일 수 있는 간단하고도 효율적인 모델을 제시한다. 핵심인 JUNOScript 는 클라이언트의 XML-RPC 를 통한 구성 정보로의 접근과 수행을 가능하게 한다. JUNOScript Server 는 태그의 분석을 통한 요청을 적절한 장비에 배분하는 역할을 하며 이 결과를 요청자에게 돌려준다.

CISCO 의 Configuration Registrar 는 CISCO IOS 네트워크 장비로 구성 파일을 자동으로 배포 시키기 위한 웹 기반 시스템이다. 각 시스템에 위치한 시스코 네트워킹 서비스(CNS) 에이전트를 동작시킨다. 장비가 처음 작동 되었을 때 초기 구성 정보를 장비에 전송하고 이를 관리한다. 이는 HTTP 로 XML 메시지를 에이전트와 통신하는 방식으로 동작된다.

위에 설명한 두 시스템을 비롯한 상용 구성 관리 시스템들은 관리에 있어서 실질적인 편리함을 제공한다. 하지만 이와 같은 구성 관리 방법은 각 사의 종속적인 데이터 구조와 통신 모델을 가진다. 이를 해결하기 위해 IETF 의 NETCONF WG 에서는 CISCO, Juniper 등의 주요 네트워크 장치 제조사들의 구성 관리 시스템을 모델로 하여 구성 관리 모델의 표준화 작업을 하고 있다.

2.3. NETCONF의 개요

이 절 에서는 현재 진행되고 있는 IETF 의 워킹 그룹인 NETCONF 의 표준화 작업에 대해서 소개한다. 그리고 NETCONF 의 관리 프로토콜 메시지를 전송하기 위한 전송 프로토콜인 SSH(Secure Shell)[16], BEEP(Block Extensible Exchange Protocol)[17], SOAP/HTTP[7]에 대해 별도로 작성된 인터넷 초안에 대해 간략히 요약한다.

2.3.1. NETCONF 프로토콜

NETCONF 는 매니저와 에이전트 간의 간소화된 RPC 메커니즘 기반의 통신 기능을 제공한다. NETCONF 프로토콜은 매니저가 네트워크 장비들의 구성 정보를 쉽게 관리할 수 있고 에이전트간의 상호 운용성을 보장하기 위하여 에이전트와 매니저 사이에 전송되는 메시지를 구조화된 XML 형식으로 정의하고 있다. 이러한 구조화된 XML 메시지를 통하여 에이전트에게 구성 정보를 가져오고, 수정하는 구성 관리 서비스를 RPC 방법으로 제공하게 된다.

- **Configuration Model**

NETCONF 의 구성정보는 running, startup, candidate 의 3 단계가 존재한다. running 단계에서는, 네트워크 장비의 동작 중인 구성정보의 현재 상태를 나타낸다. startup 단계에서는, 현재 running 상태로부터의 구성 정보가 복사된다. candidate 단계에서는, 구성 정보 변경의 히스토리를 저장해 둠으로써 구성 정보의 잘못된 수

정에 따른 복구를 가능하게 하고 네트워크 장비의 잘못된 동작을 미연에 방지하고자 한다. 구성 모델과 각 단계는 그림 2와 같다.

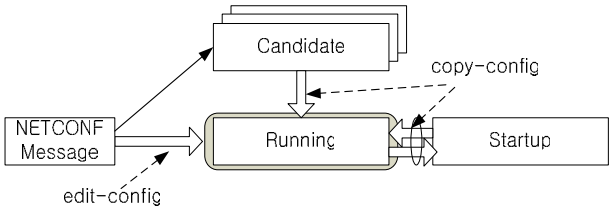


그림 2: Configuration Model

구성 관리를 수행하려는 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 4 가지 계층으로 나누어 정의함으로써 다양한 환경의 관리 요구 사항을 충족시킬 수 있게 된다.

계층	내용
Application Layer	BEEP, SSH, HTTP etc.
RPC Layer	<rpc>, <rpc-reply>, <rpc-error> etc.
Operation Layer	<get-config>,<edit-config>,<copy-config>, etc.
Content Layer	Configuration data

표 1: NETCONF 관리 프로토콜의 계층적 구조

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

2.3.2. NETCONF의 전송 프로토콜

NETCONF 의 관리 프로토콜 메시지는 어떠한 전송 프로토콜에서도 전송되어야 한다. 그러나 현재 NETCONF 에서는 전송 프로토콜로 SSH, BEEP, SOAP/HTTP 에 대해서만 고려한다. NETCONF 는 각각의 전송 프로토콜에 대해 별도의 인터넷 초안을 제안하고 있다. 이 장에서는 NETCONF 프로토콜의 Application Layer 에 해당하는 전송 프로토콜들의 각 인터넷 초안을 비교, 분석하여 장단점을 요약 정리하고자 한다.

- **SSH**

SSH 에 대해서는 3 번째 인터넷 초안까지 발표되었다. SSH 는 기존의 네트워크 장비의 관리에 있어서 주로 사용해 오던 환경이다. 따라서 기본적으로 네트워크 오퍼레이터들의 강력한 요구에 의해 제공되어야 하는 전송 프로토콜이다. 그러나 SSH 의 인터넷 초안은 구현의 간편성을 위하여 관리 수행을 위한 하나의 TCP 연결만을 제안하고 있다. SSH 의 경우 에이전트가 매니저에게 메시지를 비 동기적으로 보낼 수 있는 방법이 제공되지 않기 때문에 보고 채널 같은 경우는 유지할 수도 없다.

- **BEEP**

BEEP 은 현재 Cisco 와 Juniper Networks 와 같은 대표적인 네트워크 장비 업체에서 추천하고 있는 전송 프로토콜이며, 3 번째 인터넷 초안까지 발표되었다. BEEP 는 SSH 와 달리 NETCONF 에서 정의한 다중 채널을 지원하도록 제안하고 있다. BEEP 은 서버/클라이언트의 구조가 아닌 peer 의 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**

SOAP over HTTP 에 관련된 인터넷 초안은 4 번째까지 발표되었다. SOAP over HTTP 에서의 SOAP 은 NETCONF 의 의도에 가장 적합한 전송 프로토콜이라고 할 수 있다. 왜냐하면 SOAP 은 자체적으로 RPC 메카니즘을 제공하고 있으며, SOAP RPC 메시지는 다양한 전송 프로토콜을 지원할 수 있기 때문이다. 게다가 SOAP 을 지원하는 관련 툴들이 많이 제공되므로 시스템 구현 측면에서 다른 전송 프로토콜보다 매우 용이하다. 그러나 NETCONF 에서 전송 프로토콜을 SOAP 하나로 제한하지 못하는 것은 현재 SOAP 을 제공하고 있는 구현 환경이 전송 프로토콜로 주로 사용하는 HTTP 는 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.

3. XCMS-WS의 설계

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

3.1. XCMS-WS의 요구사항

NETCONF 인터넷 초안 및 일반적인 구성 관리 시스템에서 요구되는 기본적인 요구 사항과 이를 바탕으로 본 논문에서 제시하는 XCMS-WS 시스템을 위해 더 추가해야 할 요구사항을 알아 본다.

3.1.1. 기본적인 요구 사항

현재 NETCONF 와 일반적인 구성 관리 시스템에서 다음과 같은 요구사항을 제시하고 있다.

(1) 매니저가 에이전트에게 구성 정보를 업 로딩을 하거나 다운 로딩이 가능해야 한다. 매니저가 관리 하고 있는 모든 에이전트의 구성 정보를 중앙에서 일괄적으로 처리 할 수 있는 구조를 제공한다.
(2) 명령어를 실행 시킬 수 있는 CLI 를 제공하여야 한다.
(3) 하나의 매니저에서 여러대의 에이전트를 관리 할수 있어야 한다.
(4) 에이전트의 수정되고 있는 구성정보에 잠금을 할 수 있어야 한다. 이는 여러 매니저가 동시에 같은 구성 정보를 수정하려고 할 때 발생되는 동기화의 문제점을 해결해 준다.
(5) 매니저와 에이전트의 상호 작용을 위해 XML 형식의 메시지 구조를 정의한다. NETCONF 관리 프로토콜 메시지를 XML 로 정의하기 때문에 프로토콜의 확장성과 유연성이 용이하게 된다.

3.1.2. XCMS-WS의 요구 사항

XCMS-WS 는 기본적으로 3.1.1 장에서 언급한 기본적인 요구 사항을 만족해야 한다. XCMS-WS 에서 추가적인 고려 사항들은 아래와 같다.

(1) XPath 를 적용하여 구성 정보 모델을 수정이 가능하여야 하며, 복잡한 구성정보 수정시에는 간단한 메시지 표현이 가능해야 한다.
(2) 확장된 XPath 사용과 동시에 NETCONF 표준이 사용가능해야 한다.
(3) 웹 서비스를 이용한 메시지 전송 및 관리 구조를 제공한다.
(4) SOAP/HTTP 의 통신을 이용하며, SOAP Binding 을 이용한 메시지 전송이어야 한다.
(5) UDDI 적용을 통한 구성 장비의 발견이 가능해야 한다. 또한, 발견한 에이**전**트의 WSDL 을 통한 동적 호출이 가능해야 한다.

NETCONF ENABLE 된 장비 발견 및 관리 구조는 UDDI 를 통해 알 수 있고 에이**전**트는 모듈화 관련 아키택쳐를 통한 관리 객체의 확장이 가능하다. 디바이스의 세부 구성 정보는 NETCONF 의 초기 메시지를 통해 상호 교환이 가능함으로써 NETCONF 프로토콜과 UDDI 를 통한 관리 구조의 결합은 구성 정보의 관리 시스템으로서 적합함을 보인다.

3.2. XCMS-WS 설계

XCMS-WS 의 설계 과정은 앞서 제시한 시스템과 표준의 문제점을 반영하여 개선점을 제시한다. XPath 를 이용한 오퍼레이션의 개선 방안과 SOAP, WSDL 을 이용한 관리 정보 전송 방법과 UDDI 를 이용한 개선된 매니지먼트 방법을 포함한다.

3.2.1. 오퍼레이션 및 메시지 구조 설계

최초의 NETCONF 프로토콜[32]에서는 관리 오퍼레이션을 수행하기 위해 선택하려는 구성 정보를 나타내는데 어려움이 있다. 최초의 NETCONF 프로토콜은 선택하고 싶은 노드의 위치를 정확하게 표시할 수 없었기 때문에 선택할 구성 정보의 태그를 나타내기 위하여 그 태그의 모든 상위 부모 노드의 태그까지 서술하여야 했다. 그러나, 현재 NETCONF 프로토콜은 이러한 단점을 보완하기 위해서 하위 트리의 일부를 선택할 수 있는 하위트리 여과(Subtree Filtering) 기능이 추가되었다. 그러나 이러한 기능은 <get>, <get-config>에 대해서만 사용 가능하다.

이 문제점을 이미 XCMS-WS 의 이전 연구인 XCMS 에서 XML 데이터 노드의 위치를 명료하게 표현할 수 있는 표준 방법인 XPath 의 사용을 제안한 바 있다. XPath 를 사용하게 되면 <get-config>의 경우는 선택하고 싶은 구성 정보만을 가져올 수 있고 <edit-config>에서는 부분 수정이 가능하다.

그림 3는 XCMS 에서 제시한 edit-config operation 과 XPath 를 이용한 방법이다. edit-config-{merge|replace|delete}와 같은 표현 방식으로 operation 의 인지 단계부터 바로 서비스의

삭제, 추가, 변경 여부가 판단되는 구조이다. 또한, XPath 를 이용하여 선택되는 구성 정보의 직접 수정 작업을 가능하게 한 오퍼레이션 구조이다. 이 방법은 초기의 NETCONF 프로토콜에 비해 구성 정보의 일부분만을 지정하므로 세밀한 조작이 가능하며, 변경될 부분만 전달하므로 그 크기가 작아 지는 장점이 있다. 하지만 복잡한 구성 정보 변경을 위해서는 여러 번의 메시지의 전달을 요구하게 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>
```

그림 3: XCMS 의 <edit-config-replace>

본 논문이 제시하는 addressing 방법에서는 XPath 를 이용하며, XCMS 시스템의 문제점인 복잡한 구성 정보 수정의 표현 문제점을 config 태그를 수정하여 해결했다. 새로 제시한 방법의 예는 그림 4과 같다.

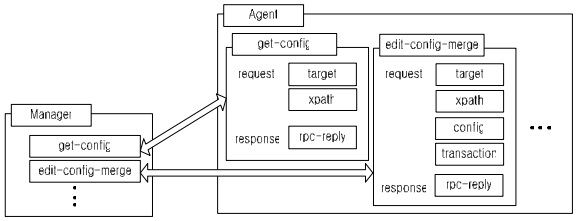
```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
         <target>
            <running/>
         </target>
         <configs>
            <config xpath
  = "//interface/name[.="Ehernet0/0"]/following::mtu
[.="1500"]/following::address" operation = "replace">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

그림 4: XCMS-WS 의 <edit-config-replace>

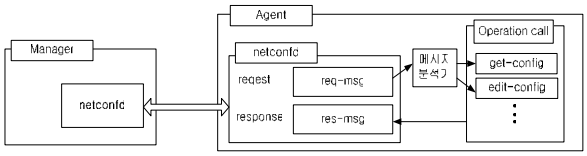
여기서 제시하는 방법은 configs 태그를 추가 하였다. 그리고 하위 노드에는 config 노드가 변경 사항의 개수만큼 추가된다. 각각의 config 노드에는 merge, replace, delete 를 위한 구성 정보가 XPath 애트리뷰트를 통해 가리키고 하위 노드에는 교체되거나 추가될 구성 정보가 표현된다. 이 방법은 XPath 를 사용하며, 하나로서 처리되어야 할 명령이 나뉨으로 생기는 트랜잭션에 대한 처리가 필요 없게 된다. 현재 IETF 에서 제시한 최근의 인터넷 초안에는 XPath 를 하위트리 여과에 사용할 수 있음을 명시하고 있으나 구체적인 사용방법은 제시를 하지 못하고 있다. 우리가 제시하는 방법이 구체적인 사용방법의 예가 될 수 있다.

3.2.2. 서비스 호출 구조 설계

XCMS-WS 에서는 NETCONF 메시지 전달을 위하여 SOAP 프로토콜을 사용한다. SOAP RPC 의 사용은 RPC 오퍼레이션의 명세가 필요하고, 각 RPC 오퍼레이션마다 WSDL 를 정의해야 한다. WSDL 정의대로 에이전트의 RPC 오퍼레이션들을 구현하게 되면, SOAP 구현 환경에서 에이전트들간에 상호 운영성이 보장 된다.



(a) XCMS 오퍼레이션 방식



(b) XCMS-WS 서비스 방식

그림 5: 서비스 방식과 오퍼레이션 방식의 비교

매니저와 에이전트간의 서비스 호출방식은 오퍼레이션 방식과 서비스 방식이 있다. 오퍼레이션 방식은 오퍼레이션 별로 서비스가 일대일로 대응되는 웹 서비스를 공개하는 방식이다. 이 방식은 이전 시스템인 XCMS 에서 채택한 바 있다. 그림 5(a)에서는 매니저 측에서 직접 오퍼레이션 별로 서비스를 호출하는 오퍼레이션 방식 서비스 호출 구조를 보여준다. 오퍼레이션 방식을 이용하면 각 오퍼레이션의 판단과 호출을 위해 매니저 측에 또 다른 하나의 계층이 필요하다. 이 방식은 전달되는 메시지가 NETCONF 프로토콜 메시지가 아니다. 반면에, 서비스 방식은 매니저가 NETCONF 메시지를 생성하고 에이전트에 메시지를 전달 하면, 에이전트 측에서 해당 메시지를 해석하여 필요한 오퍼레이션을 수행 하는 방식이다. 에이전트 단

위로 관리 정보를 가지게 되는 구성 관리 시스템 구조의 특성상 서비스 방식의 호출이 NETCONF 메시지를 전달하는 구조에 적합하다. 그림 5(b)는 XCMS-WS 의 서비스 호출 방식을 보여준다. XCMS-WS 에서의 서비스 방식 호출은 에이전트 자체를 웹 서비스화 한 netconfd 서비스를 호출한다. 그림 6의 WSDL 파일에서 보면, 에이전트를 접근하고 그 결과를 되돌려 주기 위한 하나의 request 와 response 형 만 존재한다.

해당 WSDL 파일을 통해 NETCONF 메시지를 전달하고 에이전트의 서비스 안에서 메시지 분석기를 통해 메시지를 해석하여 필요한 오퍼레이션을 재 호출하는 구조를 가진다.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="Netconfd" >
   <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>
```

그림 6: 서비스 방식의 XCMS-WS 의 WSDL

3.2.3. UDDI를 이용한 에이전트 관리 구조

NETCONF 프로토콜은 구성 관리에 있어서 장비들의 구성 정보 변경을 수행하기 위해 필요한 오퍼레이션을 정의 한 것이다. 새로운 네트워크 장치가 추가되거나 기존의 네트워크 장치에 새로운 구성정보가 추가된 경우와 같이 동적으로 구성정보 자체가 변경되는 경우에 효과적으로 적응할 수 있는 방법도 필요하다. 이러한 방법의 하나로 우리는 UDDI 를 활용하는 방안을 제시한다. 웹 서비스라고 하면 WSDL 을 이용한 서비스의 공표와 UDDI 를 통한 서비스의 발견, 또 SOAP 을 이용한 원격 서비스의 호출구조가 일반적이다.

NETCONF 에서 제시하는 프로토콜에서는 WSDL 의 정의와 SOAP 을 통한 메시지 전달에 대한 권고 사항이 정의되어 있다. XCMS-WS 에서는 이를 준수하고 추가로 UDDI 를 이용한 확장을 제시한다. XCMS-WS 에서 사용되는 에이전트 관리는 비즈니스 엔티티를 사용한다. UDDI registry 의 비즈니스 엔티티는 비즈니스 엔티티를 다른 회사에 제공하기 위해서 비즈니스 서비스를 알리는데 사용된다. 우리는 이 비즈니스 엔티티를 웹 서비스를 이용한 네트워크 관리 요소들을 등록하는 곳에 사용한다. 그림 7는 XCMS-WS 의 에이전트에 UDDI 레지스트리에 등록된 businessEntity 를 보여준다. DPNM 의 이름으로 NETCONF_AGENT 서비스가 등록되어 있음을 볼 수 있다. 각 서비스의 tModelInstanceInfo 에는 각 서비스 별 인스턴스가 등록되는데 여기의 인스턴스로 각각의 NETCONF 에이전트를 등록한다. overviewURL 에는 실제 에이전트 서비스의 wsdl 이 등록된다.

```
<?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=8f
        cc8ad7-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>
                    </tModelInstanceDetails>
                </bindingTemplate>
            </bindingTemplates>
      </businessService>
```

그림 7: DPNM UDDI 레지스트리

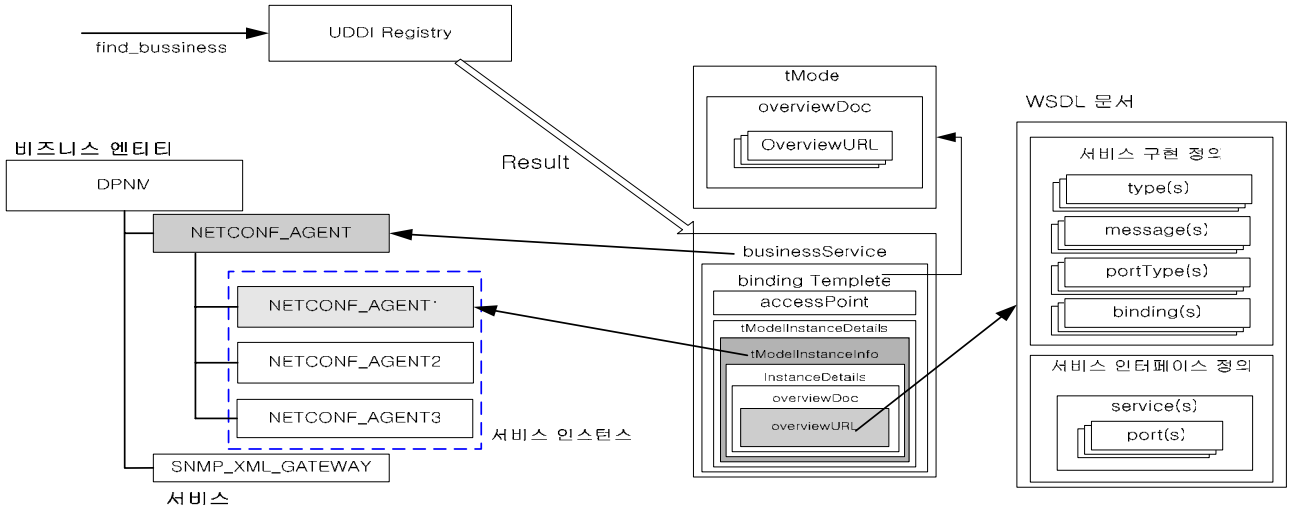


그림 8: UDDI 에 등록된 서비스

XCMS-WS 가 관리하려는 NETCONF 에이전트의 UDDI 레지스트리의 구조는 그림 8과 같다. XCMS-WS 의 NETCONF 매니저에서는 UDDI 에 등록된 NETCONF 에이전트를 찾아서 관리 가능한 장비로 등록한다. 이 과정은 find_bussiness API 를 통해 businessEntry XML 데이터를 가져온다. 여기에 포함된 bussinessService 의 NETCONF_AGENT 를 찾아 그 엔트리의 리스트를 추출한다. 엔트리의 정보는 tModelInstanceInfo 에 포함되며 overViewURL 에서 가리기는 WSDL 문서를 통해 NETCONF 에이전트 서비스의 명세를 알 수 있다.

XCMS-WS 는 로컬 인트라넷에서 에이전트 관리가 목적이므로 Private UDDI 를 사용한다. UDDI 산업 군 내의 분류가 네트워크 관리에 관한 비즈니스 분류를 갖지 않고, Public UDDI 에 등록하면 로컬의 장비를 전 인터넷에 공개하는 것이므로 보안 문제가 발생할 수도 있기 때문이다. 이는 허가된 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.

3.2. XCMS-WS구조

본 논문의 시스템인 XCMS-WS 에서 제시하는 구성 관리 시스템의 아키텍처는 크게 매니저와 에이전트, 그리고 UDDI 레지스트리의 3 단계로 구성된다. 이번 절에서는 각 단계 별로 이를 구성하는 요소를 살펴본다.

3.2.1. MANAGER

매니저는 크게 다섯 개 요소로 구성된다. 그 요소는 UI, 메시지 생성 모듈, UDDI Finder, Dynamic Invoker, SOAP 엔진이다. 이 요소를 포함하는 매니저 구조는 그림 9과 같다.

XCMS-WS 에서는 NETCONF 에서 필수로 요구하는 CLI 와 추가로 웹을 통한 GUI 환경이 제공된다. UI 를 통해 입력된 정보는 메시지 생성기를 통해 NETCONF 프로토콜 메시지로 생성된다. XCMS-WS 에서 관리 가능한 에이전트는 UDDI 에서 발견하고 사용한다. 이 방법은 UDDI Finder 모듈을 통해 이루어진다.

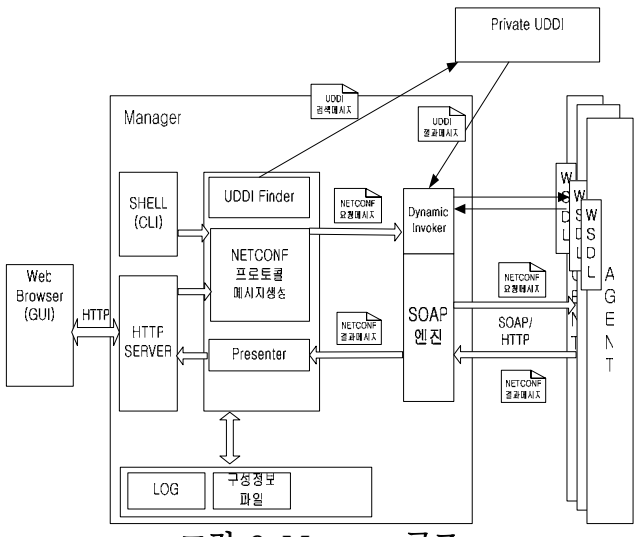


그림 9: Manager 구조

모듈에서 에이전트 정보를 가져오고 에이전트의 서비스 명세인 WSDL 의 위치를 알 수 있게 되면 WSDL 을 통한 원격 호출이 가능하게 된다. 이를 가능하게 하는 것이 Dynamic

Invoker 이다. 즉, Dynamic Invoker 는 WSDL 의 명세를 이용해 앞 과정에서 생성된 NETCONF 메시지를 에이전트로 전달하기 위한 서비스 호출 과정 처리하는 역할을 한다. 또한, 매니저에는 에이전트에게 원격으로 전달할 구성 정보 파일과 이들을 저장하는 저장 데이터 베이스 모듈 등도 포함될 수 있다. UDDI Finder 를 통해 검색된 에이전트의 WSDL 과 메시지 생성기에서 생성된 메시지는 Dynamic Invoker 와 Proxy 를 통해 에이전트의 서비스를 호출하고 메시지를 전달하게 된다. 이때 에이전트로부터 전달받은 응답 메시지는 Presenter 를 통해 GUI 로 표시된다.

3.2.2. AGENT

XCMS-WS 에서는 매니저가 요청한 서비스 수행을 위해 필요한 오퍼레이션을 바로 호출할 수 있는 RPC 방법을 사용하며, 해당 방법은 NETCONF 관리 프로토콜 메시지에 XML 태그를 사용하여 정의한다.

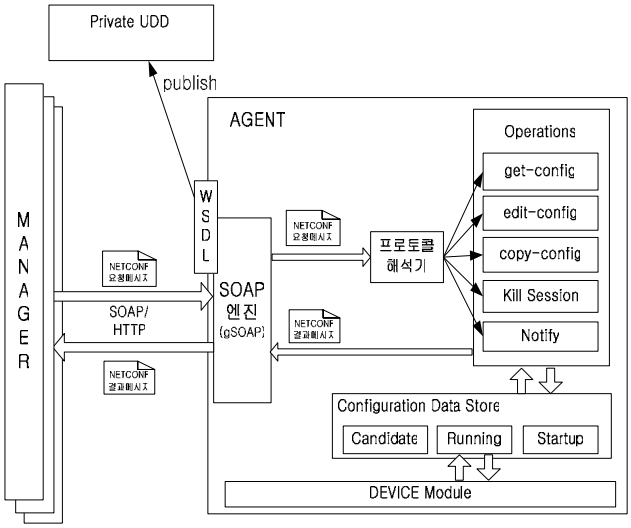


그림 10: Agent 구조

XCMS-WS 의 NETCONF 메시지는 매니저로부터 생성되어 에이전트로 전달 되는데 에이전트는 해당 메시지를 분석하여 구성 관리를 행한다. 에이전트는 메시지를 전달 받기 위한 통신 모듈인 SOAP 엔진과 메시지를 해석하기 위한 프로토콜 해석기로 구성된다. 또, 구성 관리 수행에 필요한 모든 오퍼레이션들은 실제 서비스로서 에이전트 측에 구현되어 있다. 구성 정보 모델로서 candidate, running, startup 에 해당하는 저장소가 있다. 이중 Running 은 실제 디바이스 모듈과 동기화가 되며, 해당 정보는 현재 동작 중인 상태를 나타낸다. SOAP 엔진을 이용하면 에이전트에 도착한 NETCONF 메시지를 처리 하는 과정에서 지연되는 문제점이 있다. 그러나, 메시지 큐[26]를 사용하면 이 문제를 해결할 수 있다. 메시지 큐는 여러 매니저로부터 오는 요청에 대한 처리지연을 없애, 메시지 손실의 문제를 해결할 수 있기 때문이다. 또한, 에이전트에서 관리하는 연결 정보인 Session 번호와 Session 별로 관리 되는 메시지 번호를 이용하여 lock 을 함으로써 구성정보를 동기화 할 수 있다.

프로토콜 해석기는 XML 태그로 표현된 오퍼레이션들의 이름과 오퍼레이션 수행에 필요한 파라메터들을 요청된 메시지로부터 추출한다. 추출한 정보들을 이용하여 오퍼레이션을 호출하게 된다. 오퍼레이션의 수행 과정에서 각 오퍼레이션은 구성 정보 저장소의 정보를 읽고 변경하는 등의 작업을 하게 된다. 그 수행 결과는 NETCONF 프로토콜 메시지로 매니저에게 전송한다. XCMS-WS 에서는 NETCONF 에이**전**트에서 관리할 구성 정보를 모듈로서 추가 할 수 있다. 즉, 필요한 구성 정보를 모듈화된 구조에 의해 쉽게 추가 할 수가 있으며, 이는 여러 다른 구성 정보들을 관리 하기 위한 확장성을 제공한다.

3.2.3. UDDI 레지스트리

매니저는 에이전트의 구성 정보를 자동으로 관리할 수 있어야 한다. XCMS-WS 의 UDDI 등록 구조는 에이전트의 구성 정보 이외에 관리하고 있는 에이전트의 정보가 변화 될 시 이를 재 등록하지 않고도, 매니저와 에이전트간의 구성 정보 관리 서비스 제공해 줄 수 있다. 예를 들어, 그림 11에서 보여 주 듯 에이전트의 주소 변경 시 UDDI 레지스트리에 대한 정보를 에이전트 측에서 변경해 놓으면 매니저에서는 다시 등록하는 과정 없이 자동으로 변경 내용이 반영되는 레이아웃을 제공해 준다.

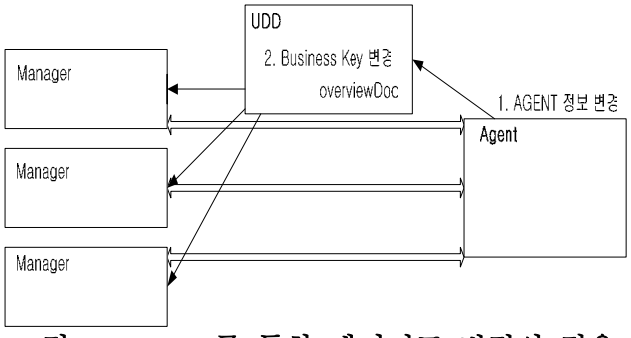


그림 11: UDDI 를 통한 에이전트 변경의 적용

4. XCMS-WS의 구현

이 장에서는 가장 최근에 발표된 NETCONF 표준을 기반으로 웹 서비스화한 구성 관리 시스템의 프로토타입 프로토타입의 웹 서비스 기반의 구성 관리 시스템 구현에 결과를 제시한다. 이 시스템의 유용성을 증명 위해 우리는 3 장에서 제시한 구조를 바탕으로 NG-MON 시스템의 구성 정보를 관리하도록 XCMS-WS 시스템을 적용 하였다.

4.1. 구현환경

- **매니저**

매니저는 다양한 XML 관련 API 및 XMLDB[31], 서블릿 컨테이너, SOAP 엔진 등 이용하기에 효율적인 Redhat LINUX 9 플랫폼(커널버전 2.4.20)을 사용하였다. 웹 서비스 및 SOAP 엔진으로 Apache[21]그룹에서 제공하고 있는 자바 환경의 AXIS[22]를 사용하였다. 자바 모듈을 웹에서 사용하고 AXIS 와 연동할 서블릿 컨테이너로는 Apache 그룹의 TOMCAT[25]을 사용하였다. 이와 같이, SOAP Over HTTP 를 위한 환경이 갖추어 지면, 프로토콜 생성 및 서비스 호출에 따른 NETCONF 프로토콜 메시지를 SOAP 프로토콜 메시지의 페이로드에 포함 시켜 보내는 과정이 가능해진다. 서비스 호출 시 WSDL 을 이용하여 Dynamic Invocation 이 수행된다. 이를 위해 WSIF[24]를 이용하였다. Dynamic Invocation 은 에이전트의 서비스가 명시된 WSDL 만으로 매니저 측에 서비스의 배치 없이 자동으로 호출 가능하도록 한다. 즉 우리 시스템은 UDDI 에서 검색된 WSDL 파일을 통해 NETCONF 에이전트의 서비스 동적 호출이 가능하다. CLI 는 gSOAP[27] 및 libxml[29]을 이용하며 자원이 부족한 환경을 위해 Shell 에서의 명령어 체계를 갖추었다.

- **에이전트**

에이전트의 경우는 컴퓨팅 자원을 최대한 아껴야 하므로 C/C++환경에서 제공하고 있는 SOAP/HTTP 소스들 중에서 가장 작고 효율적인 gSOAP 을 이용하였다. gSOAP 은 매니저의 AXIS 와 상호 운영성을 보장하므로 매니저와 에이전트간의 SOAP 메시지 전송이 가능하다. gSOAP 은 오퍼레이션을 선언한 헤더 파일을 통하여 자동으로 스텁(Stub)과 스켈리톤(Skeleton), 그리고 WSDL 을 생성한다. 또한 매니저와 에이전트 사이에 오가는 SOAP RPC 메시지도 자동으로 인코딩하여 보낸다. 에이전트와 매니저의 SOAP 서버 모듈은 gSOAP 에 내장된 Stand-alone 서버를 통하여 RPC 오퍼레이션 서비스인 웹 서비스(Web Services)를 C/C++ 환경에서 제공 한다. 에이전트에 제공되고 있는 XML 파서는 libxml 을 이용하여 구현하였다.

- **UDDI 레지스트리**

IBM 에서 제공하는 WSDK 패키지[28]에 있는 Private UDDI 레지스트리를 사용한다. Public UDDI 에 공표한다는 것은 외부 환경에 내부 장비를 공개하는 것이 되므로 보안상 Private UDDI Registry 에 등록하였다.

4.2. 작동과정의 예

우리는 XCMS-WS 시스템의 동작 과정을 보이기 위해 네트워크 트래픽 모니터링 시스템인 NG-MON[30]의 설정 파일을 관리한다.

매니저, 에이전트 구조인 XCMS-WS 는 에이전트 검색을 위해 UDDI 레지스트리를 매니저와 에이전트 사이에 배치하였다. 에이전트에서는 NG-MON 의 설정 파일을 관리 가능한 구성 정보로 가져온다.

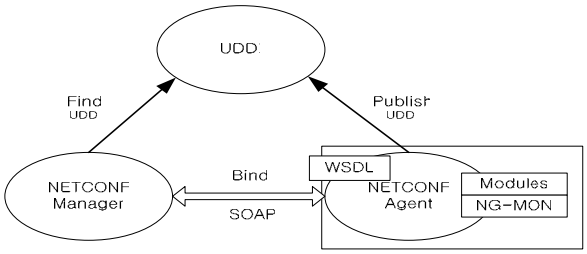


그림 12: 관리 구조의 예

그림 12은 에이전트가 NG-MON 모듈을 관리 하는 것을 보여준다. 매니저는 동일한 인터페이스를 통해 에이전트에 요청 메시지를 전달하고 에이전트는 이를 NG-MON 모듈에 전달한다. NG-MON 모듈은 NG-MON 시스템의 구성 정보를 관리 하기 위해 NG-MON 의 구성 정보 파일을 에이전트에 로딩하여 이를 동기화 시킨다. 구성 정보를 관리하는 모듈이 에이전트에 등록되면 매니저 측에서 UI 를 통해 관리가 가능하다. 이 구성 정보는 에이전트 동작 시 Running 구성 데이터 저장소와 동기화 된다.

그림 13는 매니저의 GUI 를 보여준다. GUI 의 왼편 TREE 에서는 UDDI 에 등록된 NETCONF 에이전트 및 서비스를 검색하여 표시한다. Business Entity 에 저장된 우리 웹 서비스의 정보 중 NETCONF 프로토콜을 이용하는

서비스, 즉 서비스가 동작 중인 Device 의 리스트를 가져 온다. 이는 Private UDDI 레지스트리에 등록된 정보를 우리 시스템에서 가져오는 과정에서 필요한 정보를 필터링 함으로써 가능하며, UDDI 의 내용이 바뀌면 이것이 우리 시스템에 자동으로 반영되는 동기화된 구조를 보여준다.

매니저 측에서는 CLI 와 GUI 를 통해 사용자로부터 파라메터를 입력 받는다. 메시지 생성기에서는 입력 받은 파라메터를 NETCONF 프로토콜 메시지로 생성한다. 이를 GUI 에서는 AXIS, CLI 에서는 gSOAP 를 이용하여 SOAP 메시지의 페이로드에 실어서 에이전트에 보내게 된다. SOAP 엔진은 UDDI Finder 에서 발견한 에이전트의 WSDL 파일을 이용하여 서비스의 원격 Proxy 를 생성함으로써 Dynamic Invocation 이 가능하도록 한다. 매니저는 SOAP 메시지를 에이전트에 전송함으로써 에이전트의 웹 서비스를 호출한다.

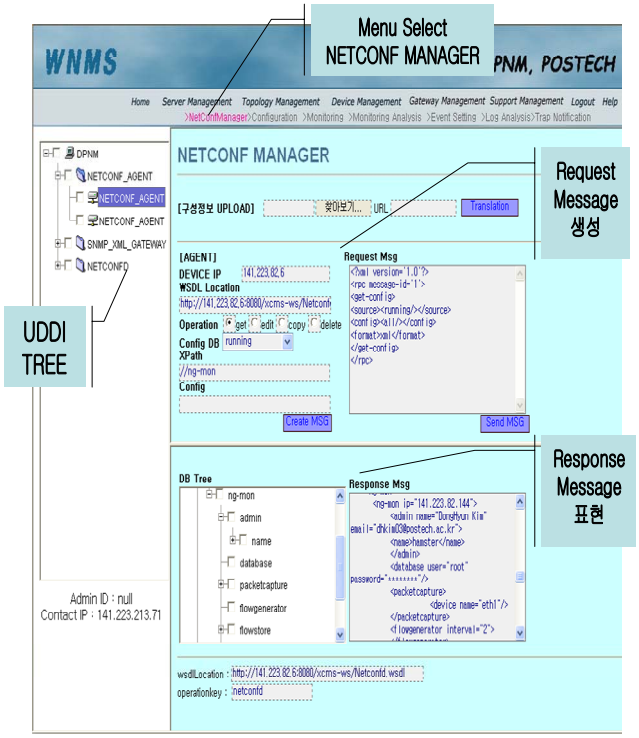


그림 13: XCMS-WS 동작 화면

에이전트 측에는 NETCONF 프로토콜 처리기가 매니저 측에서 전달된 SOAP 메시지로부터 NETCONF 메시지를 분리해 낸 후, NETCONF 프로토콜 해석기가 프로토콜을 해석하여 오퍼레이션에 해당하는 처리를 수행한다. 예를 들어 그림 14과 같이 오퍼레이션이 get-config 이고 source 구성 정보 저장소 running 이면 메모리에 로딩된 현재 상태를 나타내는 running DB 에 저장된 XML 스트림을 DOM Tree 로 생성하고 그 수행 결과를 NETCONF 메시지로 생성하여 SOAP 엔진을 통해 매니저로 전달한다.

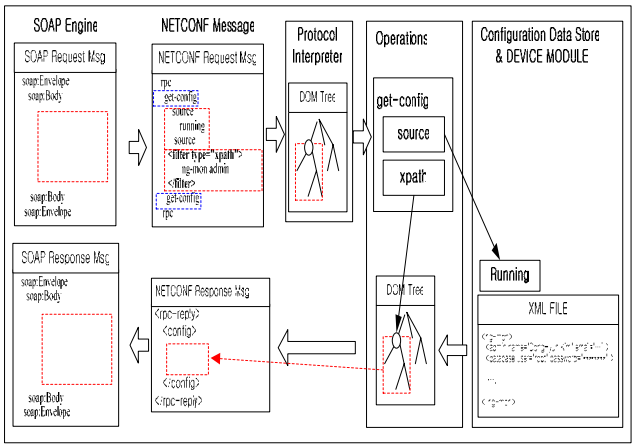


그림 14: 에이전트에서 get-config 동작과정

5. 결론 및 향후 과제

본 논문에서는 구성 관리의 문제점을 해결하기 위한 방법의 하나로 XML 기반의 NETCONF 를 언급하였다. 우리는 NETCONF 관련 인터넷 초안 및 그 관련된 시스템을 분석한 결과, 두 가지의 문제점을 발견했다. 첫째로 NETCONF 프로토콜에서 구성 정보 수정을 위한 메시지 구조가 addressing 에 있어서 비효율적이며 복잡한 구성 정보 수정 시 여러 번의 메시지 전송이 필요하다는 점이다. 둘째로 WSDL, SOAP 등의 웹 서비스 기술을 사용하나 이는 메시지 전송만을 의미해서 구성 장비 에이전트의 발견과 호출에 대한 방법이 제시되어 있는 않는 점이다.

우리는 기존의 XML 관련 툴 및 웹 서비스 관련 기술[8]들을 충분히 활용하고 앞에서 제시된 두 가지 문제점을 해결할 수 있는 개선책이 포함된 XCMS-WS 시스템을 제시하였다. 이 시스템은 XPath 를 사용한 구성 정보의 효율적인 조작 및 NETCONF 프로토콜을 이용하여 구성 정보의 효과적 통신이 가능하며 Private UDDI 를 이용한 에이전트관리 또한 효율적이다.

이번 연구가 프로토콜의 효율적 동작에 초점을 맞춘 구현 이라면 다음 연구는 활용 방안, 즉 다양한 관리 모델에 적용을 하여 보는 것이다. 즉, 상용 장비의 tModel 등록을 통한 장비별 NETCONF 에이전트 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.

참고 문헌

[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] 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.

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

[6] Enns, R., “NETCONF Configuration Protocol” draft-ietf-netconf-prot-05, Feburary 18, 2005, < http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-05.txt>.

[7] T. Goddard, “Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)”, January 2005 <http://www.ietf.org/internet-drafts/draft-ietf-netconf-soap-04.txt>.

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

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

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

[11] 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.

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

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

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

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

[16] Wasserman, M., “Using the NETCONF Configuration Protocol over Secure Shell (SSH)”, draft-ietf-Netconf-ssh-03, Feburary 20, 2005, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-ssh-03.txt>.

[17] Lear, E., Crozier, K., Enns, R., “BEEP Application Protocol Mapping for NETCONF”, draft-lear-Netconfbeep-03, November 15, 2004, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-beep-03.txt>.

[18] YENCA, a Netconf agent for Linux implemented in C, http://mady-nes.loria.fr/Software_gb.htm

[19] H. M. Choi, M. J. Choi, and J. W. Hong, “Design and Implementation of XML-based Configuration Management System for Distributed Systems”, Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, Apr. 2004, pp. 831-844.

[20] 최현미, 최미정, 홍원기, 박용석, "NETCONF 표준을 따른 XML 기반의 네트워크 구성관리 시스템", KNOM Review, Vol. 6, No. 2, February 2004, pp. 41-51

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

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

[23] Webservice Deployment Despricriptor(WSDD) Reference, http://www.os-moticweb.com/axis-wsdd/

[24] Apache XML project, Web Services Invocation Framework “WSIF”, http://ws.apache.org/wsif/

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

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

[27] 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/.

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

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

[30] Se-Hee Han, Myung-Sup Kim, Hong-Teak Ju and James W.Hong, “The Architecutre of NG-MON: A Passive Network Monitoring System”, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM2002), Montreal, Canada, October, 2002, pp. 16-27.

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

[32] 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>.

유 선 미

2004 숭실대학교, 컴퓨터학과 학사

2004 ~ 포항공과대학교, 컴퓨터공학과 석사과정

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

주 홍 택

1989.8: 한국과학기술원 전산학 학사

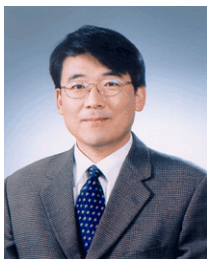
1991.8: 포항공과대학교 컴퓨터공학 석사

2002.2: 포항공과대학교, 컴퓨터공학 박사

1991.9 ~ 1997.2: 대우통신, 종합연구소, 선임연구원

2002.9 ~ 현재: 계명대학교, 정보통신학부, 조교수

<관심분야> Web-based Network Management, Network Monitoring, Mobile Device Management



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

2004 ~ 현재 IEEE ComSoc Director of On-Line Content

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