ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

다수 네트워크 관리자를 고려한 NETCONF 프로토콜의 설정 데이터 갱신 기법 개발☆

Development of Update Methods for Configuration Data of NETCONF Protocol considering Multiple Network Administrators

이 양 민[1] 차 미 양[2] 이 재 기[3*]
Yang-Min Lee Mi-Yang Cha Jae-Kee Lee

요 약

현재 이기종 네트워크를 관리하기 위해 다수 관리자가 존재하며, 이런 상황에서 효율적인 네트워크 관리를 위한 새로운 프로토콜로 NETCONF가 제안되었다. 그러나 NETCONF 프로토콜 표준은 제정 이래 지속적인 개선을 거듭해왔지만 여전히 4개의 계층에 몇 가지 문제점을 가지고 있다. 특히 다수 관리자가 존재하는 상황에서는 Operation 계층에서 문제점이 크게 부각된다. 본 논문에서는 이런 문제에 초점을 맞추어 NETCONF 4개 계층 중에 Operation 계층의 연산 효율성과 유연성을 개선하였다. 추가적으로 개선한 연산을 기반하여 장치 설정에 대한 갱신 기법의 비효율성을 개선하였다. 또한 NETCONF 표준 프로토콜에서는 제안하지 못한 Content 계층의 장비 설정 데이터 구조를 생성할 수 있는 표준적인 기법을 제시하였다. NETCONF에 세 가지 개선 기법을 적절하게 적용하고 기존 NETCONF와 제안 기법을 4가지 실험 인자로 비교하는 실험을 수행하였다. 주요 비교 인자는 네트워크 기능 유지 확률, 명령 반응 속도, 제어 패킷의 수, Content 계층에서 데이터 생성 속도라는 4가지 인자이다. 이와 같은 인자를 통해 실험을 수행 한 후 본 논문에서 제안한 기법이 기존 NETCONF 보다 우수함을 실험 결과 분석을 통해서 확인하였다.

주제어 : 관리 프로토콜, 오퍼레이션 계층, 콘텐트 계층, 연산 효율, NETCONF

ABSTRACT

Currently a number of managers exist to manage heterogeneous networks, in this situation, the NETCONF protocol for efficient network management has been proposed as a new protocol. However, the standard NETCONF protocol stack continuous improvement since the establishment but in four layers still have some problems. Especially in situations where there are multiple administrators, problems are more highlighted in operation layer. In this paper, we focus on these issues and the Operation layer has improved the efficiency and flexibility of operations among NETCONF four layers. Additionally, for the inefficiency of updates improved the device settings based on improved operation techniques. In addition, standard protocol NETCONF did not proposed content layer data structure and we propose standard technique of content layer that can generate configuration structure of devices. Improved the three techniques are applied appropriately to the NETCONF, the proposed method and the existing NETCONF was performed experiment to compare with experimental four factors. Compare key factor are four kind as maintaining the probability of network function, the reaction performance about command, the number of control packets, performance of data creation in content layer. Such factors after performing the experiment, the proposed method in this paper is superior to the existing NETCONF and there was confirmed by analysis Experimental results.

☞ keyword : Manager Protocol, Operation layer, Content layer, Efficiency of operations, NETCONF Protocol

1. 서 론

지금은 네트워크를 관리하고 유지하는데 있어서 많은 시간과 비용이 들어가는 시대이다. 그 이유는 네트워크의 필요성이 크게 증가한 것과 네트워크의 구성 방법 및 규모가 복잡하고 거대화되었기 때문이다.

1 Forhum Software CEO, Busan, 604-828, Korea
2 Professor, Dept. of Computer Engineering, Dong-A Univ. Busan, 604-714, Korea.
3 Ph.D., Dept. of Computer Engineering, Dong-A Univ. Busan, 604-714, Korea.
* Corresponding author (manson23@nate.com)
☆ 본 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임
☆ 본 논문은 2013년도 한국인터넷정보학회 춘계학술발표대회 우수논문의 확장버전임.

[Received 31 July 2013, Reviewed 9 August 2013, Accepted 25 September 2013]

이러한 네트워크를 보다 효율적으로 관리하기 위해서 기존의 SNMP로는 한계가 있어 현재는 많은 네트워크 장비에서 NETCONF 프로토콜을 네트워크 및 장비 관리용 프로토콜로 사용하고 있다. 하지만 NETCONF 프로토콜은 제정 당시부터 각 계층에서 몇 가지 문제점을 가지고 있었고, 지금까지 다양한 연구에서 수정과 보완이 이루어졌으나 여전히 문제점을 가지고 있는 것이 사실이다.

본 논문에서는 NETCONF 프로토콜의 Operation 계층과 Content 계층에 존재하는 문제점을 해결하는데 초점을 맞추었다. 특히 Operation 계층의<lock>과 <edit-config> 연산이 비효율적인 점을 개선하고, 네트워크 장비에 설정 정보 갱신 방법에 대해서도 보다 효율적인 방법을 제시하였다. 그리고 Content 계층에서는 데이터 모델링에 대한 표준이 없다는[1] 점과 모델링 기법에 대한 기본적인 방법론이 존재하지 않는 것에 착안하여 효율적인 데이터 모델링 기법도 제안하였다.

앞서 언급한 두 가지 문제점을 개선하여 현재 사용되고 있는 NETCONF에 적용할 수 있도록 실용적인 개선 기법에 대해 제시하고 이를 기존의 NETCONF와 비교하여 제안 기법의 성능이 우수함을 증명하였다.

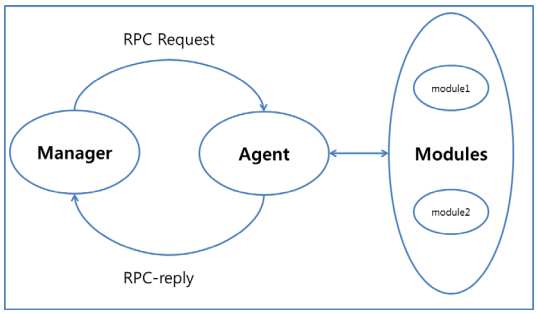
2. 관련 연구

2.1 NETCONF와 관련된 다양한 관련 연구

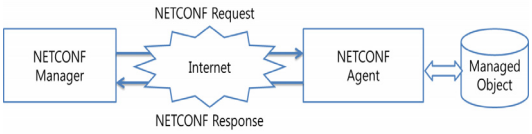
지금까지 여러 연구에서 NETCONF 프로토콜를 구현하고 그 결과를 발표하였다. 네트워크 관리 업무를 위해 YencaP라고 불리는 NETCONF 프로토콜 관리 시스템(Suite)을 제작하였는데 NETCONF 프로토콜의 여러 연구에 사용되고 있다[2].

NETCONF 프로토콜을 분석하고 NETCONF 프로토콜과 호환 가능한 IP(Internet Protocol) 공유 장비를 위해 XCMS(XML-based Configuration Management System)라는 시스템도 개발되었다[3].

또한 NETCONF 프로토콜 기반의 BUPT-NET이라고 불리는 네트워크 관리 시스템도 개발되어 있는데, 이 시스템은 매니저, 에이전트, 모듈 그룹의 세 개 파트로 구성된다[4]. (그림 1)은 BUPT-NET 시스템의 구조를 나타내고 있다. 매니저는 세션을 시작하는 클라이언트 파트로서 <rpc> 메시지를 전송하고 <rpc-reply> 메시지를 수신한다. 에이전트와 모듈들은 서버로서 역할을 하고 에이전트는 <rpc> 메시지를 수신하고 그것을 분석한 다음, 작업 달성을 위해서 필요한 모듈 내의 함수를 호출한다. 모듈들은



(그림 1) BUPT-NET 시스템의 구조
(Figure 1) BUPT-NET System Architecture



(그림 2) NETCONF 프로토콜 기반의 네트워크 관리 시스템 구조
(Figure 2) Architecture of Network Manager System based NETCONF

결과를 계산하고 그것을 에이전트에 돌려주며 에이전트는 <rpc-reply>메시지로 그것을 패키징하여 매니저에게 전송한다. 매니저와 에이전트는 SSH 프로토콜의 상단에 구축되어 있는 NETCONF 프로토콜 세션을 통해 연결되어 있다.

NETCONF 프로토콜에 기반한 네트워크 관리 구조를 개발한 연구도 있다. 이 연구의 경우 NETCONF 프로토콜 기반의 네트워크 관리 시스템 구조와 그것의 동작 요소를 기술하고 있다[5]. (그림 2)는 NETCONF 프로토콜 기반의 네트워크 관리 구조이다. (그림 2)에서 보면 NETCONF 프로토콜의 매니저는 모니터링의 요청 메시지나 설정 요청 메시지를 인터넷을 통해 전송하여 장치들을 관리한다. NETCONF 프로토콜의 에이전트는 관리 요청에 대한 응답을 돌려주는데 만약 어떤 오류가 발생하였을 경우 그것은 직접적으로 NETCONF 프로토콜의 매니저에게 전달된다. NETCONF 프로토콜의 에이전트가 설치된 객체들은 NETCONF 프로토콜 매니저에 의해 관리된다[5].

2.2 Operation 계층의 성능 개선에 관한 연구

기존 연구는 어떤 계층이 되었던 주로 처리 속도의 증가를 시키는 방법을 제시하고 있다. 특히 Operation 계층

과 RPC 계층에서 처리 속도를 증가시키는 연구들이 다수 존재하고 있다[6,7]. 다른 부류의 연구는 네트워크 회선 상에 부가적인 제어 메시지를 감소시키려는 노력과 관련된 연구들이 있다[8-10].

Operation 계층에 포함된 연산 중 <lock> 연산을 수정하거나 장비 설정과 관련된 메모리를 부분적으로 잠글 수 있게 개선하는 연구들도 있다[11-14].

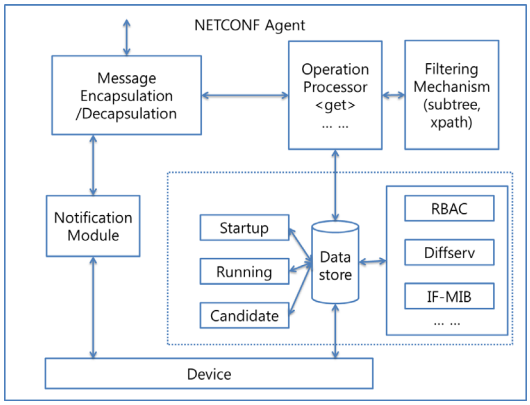
지금까지 다수의 기존 연구들은 표준 NETCONF 프로토콜이 가진 문제점을 개선하고 성능을 증가시키는데 초점을 맞추고 있다. 그리고 이와 같은 개선 기법을 적용하면 표준 NETCONF 보다는 처리 속도 등에서 보다 나은 성능을 나타낸다[15].

2.3 Content 계층의 성능 개선에 관한 연구

Content 계층은 표준을 만들기도 쉽지 않고 지금까지도 강제된 표준이 없는데 이런 문제점을 지적하는 논문은 있다[7] 그러나 이러한 연구들도 문제점을 해결할 수 있는 방법을 제시하지는 못하고 있다. 그러나 매니저와 에이전트간의 통신 전체에 있어서 데이터 모델은 매우 중요한 부분을 담당하게 된다. 따라서 데이터 모델이 중요한 이슈임에도 불구하고 현재까지 NETCONF 프로토콜에 Content 계층의 표준 모델이 제시되지 않았다. 그러므로 현재는 XML 기반 데이터 모델을 사용하고 있는데 현재 사용되는 표준인 관리 정보 베이스(MIB)를 XML 스키마 정의(Definition)로 변경할 수 있는 몇 가지 툴들을 기반으로 하여 사용되고 있다.

연구 [5]에서는 NETCONF 프로토콜 매니저가 사용자 인터페이스, 메인 프로세서, XML DB로 구성되어 있다. 사용자 인터페이스는 파일 가져오기, 객체 선택, 파라미터 설정, 응답 디스플레이라는 4가지 부분을 포함하고 있다. NETCONF 프로토콜의 에이전트는 시스템의 성능과 기능에 관계된 전체 구조에서 가장 핵심부이다. 그림3은 NETCONF 프로토콜 에이전트의 프레임워크 구조를 보여주고 있다.

NETCONF 프로토콜 에이전트에 메시지가 도착하면 XML 파서 내의 메시지 캡슐화・디캡슐화 모듈에 의해 처리될 SOAP over HTTP 및 SSL(Secure Sockets Layer) 캡슐이 분해된다. 그리고 디캡슐화된 메시지는 Operation 계층의 명령을 획득하기 위해 오퍼레이션 프로세서로 보내진다. 오퍼레이션 프로세서는 실제 현재의 장비에 대한 데이터를 저장하고 있는 데이터 저장소 내의 데이터를 편집하거나 획득한다. 오퍼레이션 프로세스는 관리하는



(그림 3) NETCONF 프로토콜 에이전트의 구조
(Figure 3) Architecture of NETCONF Agent

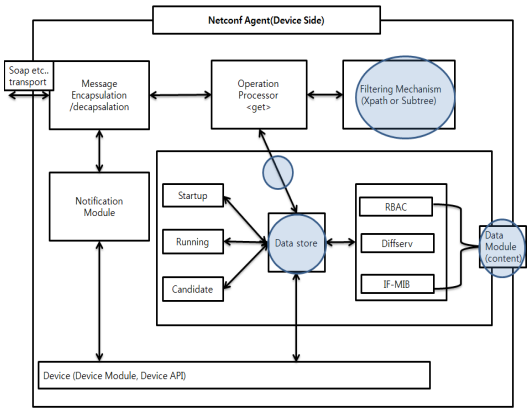
객체를 결정하기 위해 XPath 또는 서브트리 필터링 기법을 사용하며 전송되는 메시지의 유효성을 조사한다.

NETCONF 프로토콜 에이전트가 명령의 수행을 끝냈을 때 직접 응답 메시지를 생성하여 매니저에게 전송한다. 설정 데이터 저장소는 세 가지 상태(후보, 실행, 시작)의 데이터를 저장한다.

2.3 기존 연구들에 관한 고찰

앞서 기술한 연구들을 분석해 보면 성능 개선과 관련된 연구는 많이 존재하고 있다. 그러나 현재의 네트워크에서는 관리자가 다수 존재할 때가 있다. 이 경우 다수의 관리자가 하나의 장비에 대해 동시에 수정 작업을 진행할 때 이와 같은 수정 작업을 어떤 형태로 처리할 것인지에 대한 연구가 거의 없다. 다수의 관리자가 수정 작업을 효율적으로 수행하기 위해서는 <lock> 연산의 효율성이 증가되어야 한다. 기존 연구들에서는 <lock> 연산의 효율성을 증가시킬 수 있도록 개선하려는 시도도 있고[10], 장비 설정 데이터의 안정성을 위해서 <lock> 연산은 수정하지 않는 것이 좋다는 연구도 있다[12].

다수의 연구들에서 확인한 결과, <lock> 연산을 개선하려는 연구들에서는 어떤 형태의 연구이든 네트워크 장비의 Running 설정을 갱신하는 방법에 대해서 명확한 해결책을 제시하고 있지는 않다. 또한 Content 계층에 표준이 없다는 것에 대한 문제점만 언급하고 있을 뿐 효율적인 설정 정보 관리 방법을 제시하고 있지는 않다. 본 논문에서 기존 연구들을 고찰한 후 개선을 하고자 하는 부분은 (그림 4)에 나타내었다. (그림 4)는 NETCONF 프로토



(그림 4) NETCONF 프로토콜 에이전트 구조
(Figure 4) Points for improvement in NETCONF agent

콜 에이전트의 구조를 나타내고 있는데 원으로 표시한 곳이 개선하려는 부분이다.

3. 제안 프로토콜

본 논문에서는 기존 연구의 고찰에서 발견한 문제점을 해결하기 위해서 크게 3 가지 기법을 제안하였다.

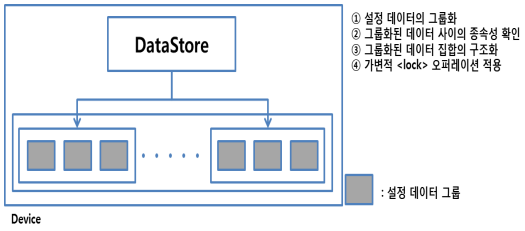
첫 번째는 관리자가 하나의 장비에 여러 가지 명령어를 수행하거나 다수의 관리자가 하나의 장비에 접근할 때 문제가 될 수 있는 <lock> 연산의 효율성을 개선하는 것이다. 두 번째는 다수의 관리자가 하나의 장비에 접근하였을 때 장비의 설정 정보를 개선하기 위한 XML 문서 갱신 기법과 관련된 내용이다. 마지막으로 Content 계층에서의 설정 정보 생성 방법에 대한 제안과 생성된 데이터의 유지 방법론이다.

3.1 lock 연산의 효율성 개선 기법

본 논문에서는 표준 NETCONF에서 지원하는 <lock> 연산의 유연성을 부여할 수 있도록 기능을 개선하였다. 일반적인 <lock> 연산의 경우 여러 관리자가 하나의 장비에 명령을 내리고 동시 연산이 수행될 경우에 발생할 수 있는 데이터 불일치를 예방하기 위해서 장비의 DataStore 전체를 잠그게 된다. 장비 설정 데이터의 안전한 변경 보장이라는 관점에서 보면 이러한 방식이 좋지만 장비의 설정 변경 연산 속도와 편의성면에서 보면 DataStore의 특정 일부분만을 잠글 수 있는 연산이 필요하다.

본 논문에서는 데이터 종속성 선행 검증 기법을 적용하였다. 이 기법은 변경할 데이터를 미리 그룹화하고 분류할 수 있도록 한 다음, 이 그룹화 된 데이터들 간의 종속성도 미리 규칙을 정하여 구분한다. 그리고 그룹화 한 데이터 집합을 계층적으로 구조화 하여 종속성에 문제가 생기지 않도록 한다. 마지막으로 각 데이터 집합에 대해 사용자가 원하는 연산별로 가변적인 <lock-p> 연산을 선택적으로 적용할 수 있도록 하였다.

(그림 5)는 DataStore에 가변 <lock-p> 연산을 적용하는 형태를 도식화한 것이다. 데이터 종속성과 관련된 것은 트리 형태로 구분이 되고 각각의 사각형은 데이터를 미리 분류하여 그룹화한 집합을 의미한다. 이와 같은 구조를 이용하여 DataStore 전체를 잠글 필요가 없이 부분적으로 <lock> 연산을 적용할 수 있고, 연산 효율의 증가와 데이터 무결성이 동시에 보장된다.

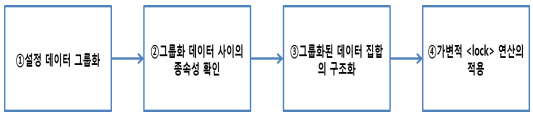


(그림 5) 가변 〈lock-p〉오퍼레이션 적용 기법
(Figure 5) Improved Method with 〈lock-p〉 Operation

본 논문에서 사용하는 데이터 종속성 선행 검증 기법은 NETCONF 프로토콜를 사용하는 네트워크 장비들 사이에서만 가능하다. 설정 데이터를 그룹화하기 위해서 XML Path Language (XPath: XML 경로 언어)를 사용한다. XPath는 마크 업 언어 XML 기반 문서의 특정 부분을 지정하거나 부모-자식(Parent-Child) 관계 등을 지정하는 언어 구문이다. XPath를 적용하게 되면 장비의 설정 데이터를 설정하는 시점에서 관리 대상의 위치나 다른 대상과의 관계 등을 미리 지정할 수 있다. 즉 관리 대상이 XML 문서 내에서 어떤 위치를 가져야 하는지와 다른 대상과 어떤 관계를 가지는지에 대한 정보를 구조화 된 데이터로 모델링 할 수 있다는 뜻이다. 그러므로 본 논문에서는 XPath를 사용하여 장비의 설정 데이터를 모델링하는데 이를 통해서 설정 데이터의 그룹화를 바로 수행할 수 있다.

이후 부모-자식 관계 또는 부모-자식-형제 관계까지를 활용하여 그룹화 데이터 사이의 종속성도 확인이 가능하다. 부모-자식 관계에 있는 데이터는 종속성이 있는 것으

로 판단하며, 가장 먼저 확인된 것이 제일 말단의 자식이든 루트이든 상관은 없다. 어느 한 방향으로 수직적으로 연결되어 있다면 종속적인 데이터이므로 종속성을 확인할 수 있다. 종속성이 확인되면 현재 수정하고자 하는 관리자와 관리하고자 하는 데이터에 따라 각각의 구룹화된 데이터 집합을 구조화 할 수 있다. 관리자가 접근하고 있는 데이터를 중심으로 수직적으로 연결된 것들을 하나의 데이터 집합으로 하면 되고 수평으로 연결된 형제 관계의 데이터는 독립적인 데이터 집합으로 구분할 수 있다.



(그림 6) 가변 〈lock-p〉오퍼레이션 적용 순서
(Figure 6) 〈lock-p〉 Operation Process Phase

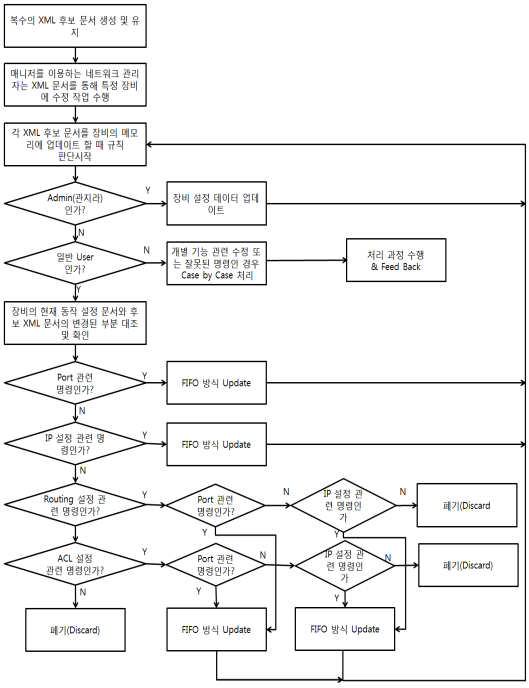
마지막으로 그룹화된 데이터 집합에 대해서 <lock> 연산을 부분적으로 적용하면 늘 상황에 적합한 가변적 <lock-p> 연산을 적용할 수 있다. (그림 6)은 DataStore의 일부분만을 잠그는 가변 <lock-p> 연산의 수행 순서이다.

3.2 Operation 계층의 설정 정보 갱신 기법

이기종 장비로 구성된 네트워크에서 여러 명의 관리자가 하나의 장비에 접근하여 설정을 변경할 때 사용할 수 있는 방법은 동작 설정을 업데이트 할 수 있는 후보(Candidate)를 복수 개로 유지하는 것이다[13].

하나의 장비에서 실제 동작은 동작 설정 데이트를 이용하기 때문에 동작 설정을 수정할 후보 설정을 여러 개 두고 본 논문에서 제안하는 규칙에 따라 후보들 중 적합한 문서를 이용하여 동작 설정을 갱신하도록 한다. 후보 문서는 XML로 작성한 문서이며 이들은 장비의 에이전트가 관리하도록 하고 메모리의 허용 범위 이내에서 네트워크 관리자 숫자만큼 생성하고 유지할 수 있다. (그림 7)은 후보 문서 갱신 규칙을 기술한 것이다. 가장 우선순위가 되는 것은 관리자의 권한을 나타내는 플래그이며, 다음 순위는 장비에 대한 변경 범위를 나타내는 플래그이다.

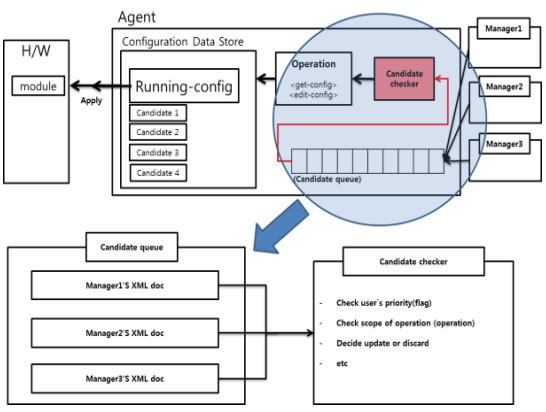
Operation 계층에서 가지고 있는 문제점인 <lock> 연산의 비효율성을 해결하기 위해서 연산을 새롭게 정의하는 방법이 요구된다. 그러나 실제로 연산을 재정의 하는 방법보다는 RPC 명령을 가지고 있는 XML 문서를 네트워크 관리자의 숫자만큼 둔다. 그리고 각각의 관리자들은 자기가 보고 있는 문서를 동시에 고치는 것은 가능하게 하고 실제로 이것을 동작 설정(Running-Config)에 업데이트 할 때 어느 것을 적용할 것인지에 대한 규칙을 정의하는 것이 효율적이다. <lock> 연산을 수행할 때는 에이전트에 도착한 세션 번호와 사용자의 권한 플래그를 이용해서 lock의 범위를 결정하고 수정의 가능 유무를 결정할 수 있다. 도착한 메시지의 세션 번호는 다른 변경 명령과 구분하는 역할을 하고 사용자의 권한 플래그는 우선 순위를 결정하는데 사용된다.



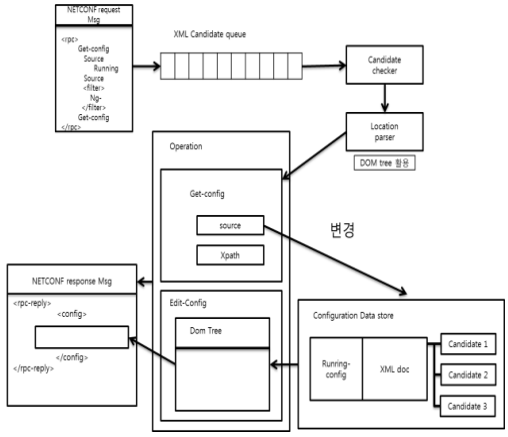
(그림 7) 장비 설정 정보를 위한 XML 문서 갱신 규칙
(Figure 7) Rule set of Updating XML Document for device Configuration Information

(그림 8)은 에이전트 내부에서 장비 설정 XML 문서 갱신 규칙이 적용되어야 하는 부분을 보여주고 있다. 후보 문서 체크를 하는 모듈이 핵심적인 역할을 하며 그 앞의 큐는 여러 관리자로부터의 문서를 저장하는 역할을 한다.

(그림 9)는 장비의 설정 변경을 위해 NETCONF 메시지가 처리되는 순서를 도식화한 것이다. XML 문서로 작성된 메시지는 후보 문서를 저장하는 큐에서 대기하다가 후보 문서 체크 모듈을 통과한다. 이때 본 논문에서 제안한 갱신 규칙이 적용되고 체크를 통과한 후보 문서는 장



(그림 8) 에이전트 내부에서 후보 체크 모듈 위치
(Figure 8) Location of Candidate check module in a agent



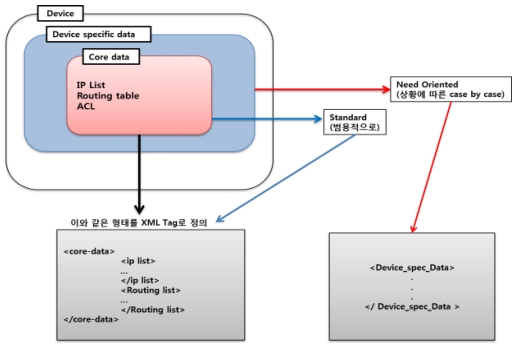
(그림 9) NETCONF 메시지의 처리 순서
(Figure 9) The Order of NETCONF Message Processing

비의 어떤 설정을 수정해야 하는지를 확인하는 위치 파서(Location Parser)를 통과하여 설정 데이터 스토어에 저장된다. 이후 규칙에 따라 동작 설정을 업데이트하게 되면 그 결과가 관리자에게 피드백 되는 것으로 처리가 종료된다.

3.3 Content 계층의 설정 생성 및 유지 방법론

Content 계층의 표준화 문제에 대한 해결 방법은 완벽한 해결 방법이 존재하지 않는다. 이유는 장비 생산 업체마다 장비 자체의 고유 기능과 운영체제 버전이 존재하는데 이것을 모두 포괄하는 설정 데이터 구조를 만드는 것은 불가능하기 때문이다. 그러나 핵심이 되는 라우팅 테이블, 포트 설정 명령, ACL 작성과 같은 부분은 동일하기 때문에 동일한 부분을 중심으로 하는 XML 문서 기반 데이터 구조를 만들었다. 데이터 구조의 기본 형태는 Cisco 장비에서 사용하는 것을 기본으로 하였다.

공통화 할 수 없는 부분은 그대로 두고 관리자를 이용하여 수정 작업을 시행할 때 변경할 수 없는 부분이나 데이터 구조가 다른 부분은 설정 데이터의 갱신이 불가능하도록 하고 그러한 결과를 정확하게 피드백 하는 방식을 사용하였다. (그림 10)은 이와 같은 구조를 만들기 위한 XML 문서 생성 규칙과 태그의 예시를 나타낸 것이다. 필요에 따라 여러 가지 새로운 태그 생성이 가능하지만 핵심적인 데이터와 장비 종속적인 데이터를 구분하는 것으로 충분하다. 이와 같은 방법으로 Content 계층의 데이터를 관리하면 보다 효율적이고 비교적 표준에 가까운 관리가 가능하다.



(그림 10) Content 계층 데이터 구조 생성 예시
(Figure 10) Example of Content layer Data Structure Creation

4. 실험 및 분석

4.1 실험 환경

실험은 장비 설정 메모리 갱신 규칙을 알고리즘으로 구현하고 그 처리 과정을 큐 이론에 기반한 C 언어로 구현하였다. 단일 컴퓨터인 로컬 환경에서 실험을 수행하였고, 관리자는 수는 10명으로 제한하였다. 그 중 최고 권한 관리자는 1명이며, 나머지 관리자는 동등한 권한으로 설정하였다. 기존의 NETCON로 처리하는 방식과 제안한 기법이 처리하는 방식을 순수 알고리즘으로 비교한 결과를 수치화하여 결과를 도출하였다.

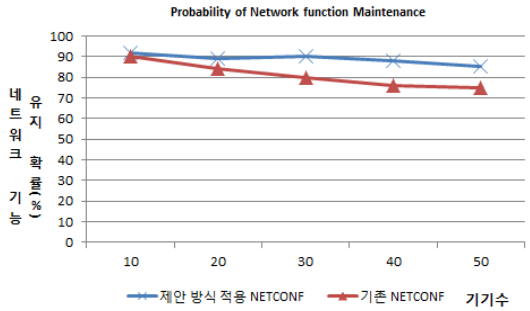
비교 인자는 네트워크 기능 유지 확률, 제어 패킷의 개수, 명령의 반응 속도, 데이터 생성 속도이다.

4.2 네트워크 기능 유지 확률과 관리자 수에 따른 명령 반응 속도 비교

Operation 계층의 기능에서는 <get-conifg-m>, <lock-p> 연산의 범위를 조절하여 효율성을 증가시키는 기법과 다수의 매니저가 있을 때 장비의 설정 정보를 효율적으로 업데이트 하는 기법을 중심으로 개선하였다. 오늘날 네트워크에서 관리자의 수가 여러 명 있는 경우는 빈번하며, 하나의 장비에 대해서 동시에 작업을 수행할 확률은 정확하게 특정할 수 없지만 빈번하게 발생할 수 있다. 따라서 적용 범위를 수정한 <get-conifg-m>, <lock-p> 연산의 효율성에 대한 검증과 장비 설정 데이터의 갱신 효율성에 대한 실험을 각각 수행하였다. 그리고 각 실험에 대해서 네트워크 오버헤드를 확인하는 실험을 함께 수행하였다.

네트워크 장비에서 조회할 데이터를 위한 <get-config-m> 연산의 범위를 조절하고, 장비 설정 데이터의 종속성 검사를 통해 가변 <lock-p> 연산을 적용하는 기법이 Operation 계층의 효율성 증가를 위해 우선적으로 적용되었다. 이 기법의 성능 비교를 위해서 네트워크 기능 유지 확률을 비교하였다. 즉, 장비의 데이터 조회를 수행하다 보면 회선의 문제가 아닌 장비의 문제로 네트워크가 기능하지 못하는 상태가 발생하는데 이런 경우가 어느 정도 발생하는지를 확인하는 것이다. 또한 <lock> 연산이 장비 전체의 메모리를 잠그게 될 경우에도 네트워크 장비가 제대로 기능하지 못하는 경우가 있다. 이를 동시에 확인할 수 있는 실험이 네트워크의 기능 유지 확률이다.

기존 NETCONF 프로토콜의 경우 기기수가 늘어나게 되면 <get-config>와<get> 연산이 작동하여 많은 데이터들



(그림 11) 네트워크 기능 유지 확률
(Figure 11) Probability of Network Function Maintenance

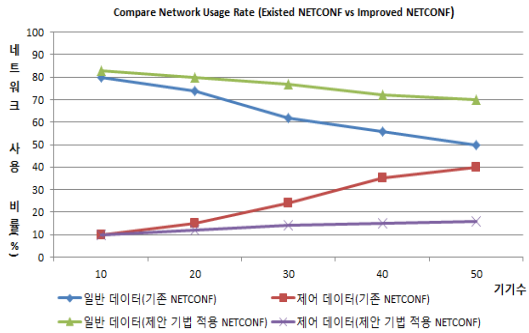
이 전송되며 경우에 따라서 <get-config> 연산이 동작하면서 네트워크 장비의 상태 데이터를 무시하기 때문에 네트워크의 기능이 일부 마비되는 현상이 간혹 나타난다. 예를 들면 라우터의 포트에 대한 설정이 잘못된 경우이다. <lock> 연산이 네트워크 장비의 메모리 전체를 잠그고 설정 데이터를 수정한 후 메모리가 전체적으로 갱신될 경우에도 비슷한 문제가 발생할 수 있다.

(그림 11)에 네트워크 기능 유지 확률을 나타내었는데 제안한 NETCONF 프로토콜의 경우 <get-config-m> 연산이 동작하고 설정 데이터와 상태 데이터를 우선 비교하여 서로 다른 부분만을 전송할 수 있기 때문에 네트워크 부하도 작고 우선순위가 높은 데이터를 부분적으로 조회할 수 있기 때문에 네트워크 기능의 유지에 있어 기존 방식 보다는 상대적으로 우수하다. 또한 본 논문의 <lock-p> 연산은 종속성 검사를 위해서 XPath를 적용하여 부분적인 <lock> 연산이 가능하기 때문에 장비의 설정 데이터에 대해서 전체적으로 끼치는 영향이 적다. 이를 수치화 시켜보면 다음과 같다.

<lock> 평균처리 시간 (1 KByte 기준) : 0.3초 미만
<lock-p> 평균처리 시간(1 KByte 기준): 0.5초 이상
<lock>의 DataStore 잠그기에 의한 라우터 정지 확률 (장비 수 최대 50, 관리자10명, 포트 수정 기준): 9%이상
<lock-p>의 DataStore 잠그기에 의한 라우터 정지 확률 (장비 수 최대 50, 관리자10명, 포트 수정 기준): 5%미만
<get-cofig>의 장비 상대 데이터 무시에 의한 정보 재 조회 및 라우터 오동작 확률 : 5% 이상
<get-cofig-m>의 장비 상대 데이터 무시에 의한 정보 재 조회 및 라우터 오동작 확률 : 2% 미만

본 논문에서 제안한 <lock-p> 연산은 종속성 선행 검사를 XPath를 활용하여 수행하기 때문에 데이터 처리 속도는 분명히 느리다. 그러나 종속성 선행 검사를 통해서 DataStore를 부분적으로 잠그는 기능이 가능하기 때문에 네트워크 관리자 10명과 장비의 수 최대 50대를 가정하였을 때 라우터의 기능 정지 확률이 최대 5%이다. 기존 NETCONF는 9%를 조금 넘는 라우터 정지 확률이 나오는데 이것은 네트워크 관리자들의 명령 중복이나 동일 장비에 대한 접근 시에 DataStore 전체가 잠기기 때문이다. 유사하게 <get-config>는 장비의 상태 데이터를 무시하기 때문에 라우터가 오동작하거나 경우에 따라 정보를 다시 조회해야 한다. 이 확률이 <get-config>의 경우 5%이상 발생하는데 <get-confg-m>은 상태 데이터와 설정 데이터에 대한 비교를 먼저 수행하여 사실상 장비의 상태 데이터를 무시하지 않는다. 그러나 내부적으로 비교 연산 오류

등에 의해 약 2% 정도의 오동작 확률은 존재하며 따라서 전체적으로는 제안한 NETCONF 프로토콜이 기존 NETCONF 프로토콜보다 평균적으로 약 7.7% 기능 유지 확률이 높다.



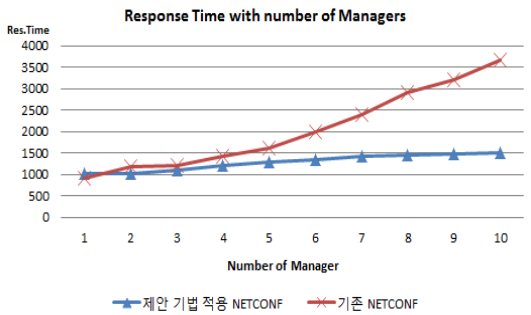
(그림 12) 제어 데이터의 네트워크 사용 비율
(Figure 12) Usage Rate of Control Data in Network

(그림 12)는 DataStore에 대한 <get-config-m> 연산과 가변 <lock-p> 연산이 연산 자체의 성능 향상을 위해서 네트워크를 이용하는 제어 데이터가 일반 데이터 대비하여 네트워크를 사용하는 비율을 확인하는 실험의 결과이다. 즉, 네트워크 오버헤드를 확인하기 위한 실험이다. 그림 12에 제어 데이터의 네트워크 사용 비율을 나타내었는데 제어할 네트워크 장비의 수가 증가할수록 제어 데이터의 비율이 높아지는 것은 기존 NETCONF 프로토콜과 제안한 NETCONF 프로토콜이 모두 동일하다. 그러나 제어 데이터의 증가 비율을 확인해보면 제안한 NETCONF 프로토콜에서의 기법은 네트워크 장비의 수가 늘어나도 제어 데이터의 비율이 급격하게 늘어나지 않는다. 반대로 기존 NETCONF 프로토콜은 장비의 증가와 더불어서 제어 데이터가 급격하게 늘어나고 네트워크 대역폭의 사용 비율이 높아지는 것을 알 수 있다.

분석 근거는 앞서 실험에서의 근거를 활용할 수 있다. 가장 주요 근거는 <lock-p>연산의 경우 내부적으로 종속성 선행 검사 등을 수행하지만 네트워크로 제어 데이터를 내보내는 것은 줄어든다는 것이다. 장비의 DataStore가 잠겨서 네트워크 관리자들이 재접근을 한다든지 다시 명령을 보내는 수가 감소하고 더불어 <get-config-m>에 의해서 장비의 상태 정보를 재조회하는 수도 크게 감소한다. 따라서 기존 NETCONF 프로토콜에 비해 제안한 NETCONF 프로토콜에서 사용하는 기법이 평균적으로 제어 데이터를 약 11.4% 적게 사용한다.

다음으로는 다수의 관리자가 존재하고 이 때 다수의 네트워크 장비에 대해서 설정 데이터를 갱신하고자 할 때 효율적으로 장비의 설정 정보를 갱신하는 기법에 대한 실험이다.

다수의 네트워크 관리자가 동시에 장비 설정이 저장된 메모리에 접근하고 특정 변경 명령을 내렸을 경우 가능한 빠른 시간 내에 연산 결과가 적용되고 그 결과를 확인할 수 있어야 한다. 따라서 주요 비교 인자는 명령의 반응 속도와 실제로 발생되는 제어 패킷의 개수이다. 명령의 반응 속도는 다수의 관리자가 장비 설정 변경을 수행하고 응답을 수신하는 시간을 측정한 것으로 중요한 인자이다. 제어 패킷의 개수는 네트워크 오버헤드를 측정한 것으로 보조적인 인자이다.



(그림 13) 관리자 수 증가에 따른 명령 반응 속도
(Figure 13) Command response time with increasing number of managers

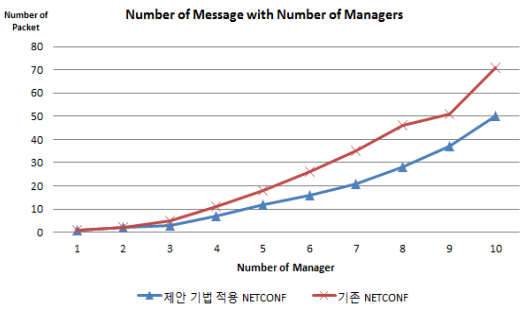
(그림 13)은 관리자 수에 따른 명령 반응 속도를 비교한 것이다. (그림 13)을 보면 관리자 수 증가에 따라 각 관리자가 내린 명령이 수행되고 반응이 돌아오는 시간을 그래프로 나타내었는데, 실험 결과는 관리자 수가 증가하더라도 본 논문에서 제안한 NETCONF 프로토콜이 기존 NETCONF 프로토콜보다 명령 반응 속도가 빠름을 보여주고 있다.

<lock> 평균처리 시간 (1 KByte 기준) : 0.3초 미만
<lock-p> 평균처리 시간(1 KByte 기준): 0.5초 이상
<lock>의 DataStore 전체 locking에 의한 네트워크 관리자의 재 명령 시간: (장비 수 최대 50, 관리자10명, 포트 수정 기준): 3.5초 이상
<lock-p>의 DataStore 부분 locking에 의한 네트워크 관리자의 재 명령 시간 (장비 수 최대 50, 관리자10명, 포트 수정 기준): 1초 이상

이 실험의 분석 근거는 매우 단순하다. 제안한 기법인 <lock-p>의 경우 종속성 검사를 위해 내부 프로세싱 시간이 필요하며 <get-config-m>도 장비의 상태 정보와 설정 정보를 비교하기 위한 시간이 필요하다. 그러나 일반적으로 내부 컴퓨팅 시간 보다 네트워크를 통한 명령 전달과 결과 보고를 받는 시간이 훨씬 많이 소모된다. 근거로 제시한 값처럼 네트워크 관리자들이 DataSotre 전체에 대한 locking으로 인해 다시 명령을 내리는 시간과 이에 대한 반응 속도를 측정하면 <lock>과 <lock-p>의 연산 속도 차이는 무시할만한 수준이다. 따라서 제안한 NETCONF 프로토콜은 설정 XML 문서에 대한 후보를 다수로 두고 경우에 따라 병렬적으로 장비의 설정 정보를 수정할 수 있기 때문에 다른 관리자의 접근이 끝날 때까지 대기해야 하는 기존 NETCONF 프로토콜보다는 반응 속도가 빠른 것이다. 다만 반응 속도는 빠를지라도 장비에 대한 설정 데이터를 XML 문서로 다수 유지해야 하고 갱신 규칙 알고리즘을 실행해야하기 때문에 컴퓨팅 시간이 다소 필요한 것도 사실이다.

본 논문에서는 갱신 규칙 알고리즘을 적용하는 컴퓨팅 시간에 대한 실험은 수행하지 않았다. 컴퓨팅 시간은 네트워크 시뮬레이션이나 실제 네트워크상에서는 확실한 오버헤드로 작용할 가능성이 높지만 본 논문에서처럼 알고리즘만을 시뮬레이션 한 경우에는 크게 의미를 두지 않았기 때문에 실험에서 제외하였다. 대신 다른 문제가 될 수 있는 네트워크 오버헤드에 관해서 실험을 수행하였는데 이것은 관리자 수의 증가에 따른 제어 패킷 개수의 증가 상태를 확인하는 것을 실험하였다.

(그림 14)는 관리자 수의 증가에 따른 순수 제어용 패킷 수의 증가 경향을 확인시켜 준다. 명령어를 담은 패킷을 관리자마다 1개식 주기적으로 생성 시킨다는 조건하에서 실험하였다. 이때 기존 NETCONF 프로토콜은 관리자 중에 명령이 실행되지 않아 그 응답을 수행하기 위해서 다시 제어 패킷을 네트워크로 보내는 횟수가 증가한다. 이런 현상은 관리자 수가 증가할수록 현저하게 증가한다.



(그림 14) 관리자 수 증가에 따른 제어 패킷 수
(Figure 14) The Number of Control Messages in increasing Number of Mangers

본 논문의 제안한 NETCONF 프로토콜에서는 관리자들이 피드백을 받아 명령어가 실행되지 않은 이유 등을 확인할 수 있다. 또한 병행 처리가 되는 경우도 많기 때문에 관리자 수가 증가해도 제어 패킷의 수가 크게 증가하지는 않는다.

실험 결과를 보면 네트워크상으로 발송되는 제어 패킷의 수도 제안한 NETCONF 프로토콜이 우수한 결과를 보여주는데 이것은 관리자의 수가 최소 7명에서 10명 이상되어야 의미 있는 성능 차이를 보인다. 즉 관리자 수가 그 이하이고 관리 대상이 될 네트워크 장비가 충분하지 않을 경우에는 필요 없는 네트워크 오버헤드, 컴퓨팅 시간, 메모리 낭비가 발생할 가능성이 있다.

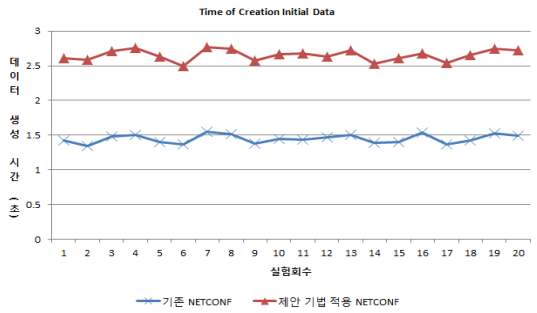
4.3 Content 계층에서 데이터 생성 속도 비교

본 논문에서 제안한 기법을 적용하여 Content 계층에서 데이터를 생성할 때의 속도를 비교하였다. 실험 회수는 20회이며 초기에 데이터를 생성할 때와 이미 생성되어 있는 데이터에 추가적인 확장 데이터를 생성할 때의 속도를 구분하여 실험하였다.

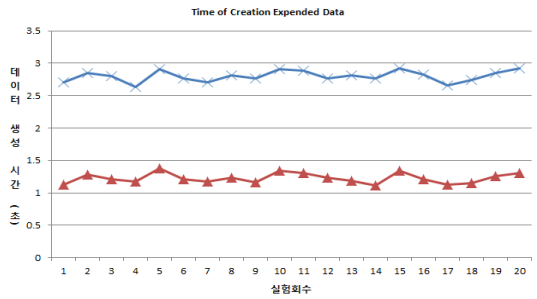
초기 데이터 생성 속도는 (그림 15)에 나타내었는데 초기 데이터 생성시에는 기존 NETCONF가 제안 기법을 적용한 경우보다 평균적으로 약 1.2초 빠르다.

본 실험은 XML 문서 생성을 기초로 하여 Content 계층 데이터 생성 알고리즘을 적용한 것이기 때문에 표준적인 수학적 근거는 별도로 없다. 다만 제안 기법의 경우 Content 계층 데이터 생성을 위해 일반 데이터와 장비 종속적 데이터를 구분할 필요가 있기 때문에 이를 위한 내부 프로세싱 시간이 1.2초 정도 더 추가적으로 소모 된다. 새로운 파일 생성 시간과 내부적 프로세싱을 합친 시간으로 계산하면 이와 같은 결과를 얻을 수 있다.

확장 데이터를 생성할 때는 제안 기법의 기존의 NETCONF에 비해서 평균 1.57초가 빠른데 이미 데이터를 생성할 때 일반 데이터와 확장 데이터가 구분이 되어 생성되어 있고, 새로 생성할 데이터가 포함되어야 할 위치를 미리 결정할 수 있으며 데이터를 생성하는 방법도 미리 설정이 되어 있기 때문에 초기 데이터를 생성할 때와는 다른 결과가 나타난다. 즉 기존 NETCONF의 경우



(그림 15) 초기 데이터 생성 시간
(Figure 15) Creation Time of Initial Data



(그림 16) 확장 데이터 생성 시간
(Figure 16) Creation Time of Expended Data

추가적인 데이터를 생성할 때나 확장 데이터를 삽입할 때에도 처음 데이터를 생성하는 것과 같은 과정을 거쳐야 한다. 그러나 본 논문에서 제안한 기법의 경우 처음과 동일한 과정이 필요 없고 정보를 삽입할 구역이 파일 내에 만들어져 있기 때문에 (그림 16)과 같은 결과를 나타내는 것으로 분석된다.

5. 결론 및 향후 과제

다수의 네트워크 관리자가 존재할 정도로 현대의 네트워크 관리는 복잡하고 시간, 인력 집중적인 업무이다. 네트워크 관리를 원활하게하기 위해 NETCONF 프로토콜이 제안 되었지만 각 계층에서 해결되지 않은 문제점을 가지고 있다. 본 논문에서는 여러 가지 문제점 중에서도 다른 연구들에서 많이 다루고 있지 않는 다수의 관리자가 존재할 때의 장비 설정 데이터의 갱신 기법을 제안하였다. 이를 보조하기 위해서 Operation 계층에 존재하는 연산의 효율성에 대한 개선 연구도 수행하였다. 또한 Content 계층에서 데이터 구조를 생성하는 방법도 표준이 없어 기존 연구에서 충분히 다루고 있지 않다. 본 논문에서는 이러한 부분에도 Content 계층의 데이터를 핵심 데이터와 장비 종속적 데이터로 구분하고 XML 태그를 재정의 하는 방법을 제안하였다.

알고리즘 중심의 실험을 수행한 결과 기존 NETCONF 보다 본 논문에서 제안한 개선 기법을 적용한 NETCONF 프로토콜이 네트워크 기능 유지 확률에서는 약 7.7% 이상 높음을 확인하였다. 명령 반응 속도 또한 기존 NETCONF 보다 빠름을 확인할 수 있었고 이에 반해 제어 패킷 생성 수는 기존 NETCONF 대비 크게 증가하지 않음을 확인할 수 있었다. Content 계층의 데이터 생성 시간 또한 초기 데이터 생성 시에는 제안 기법이 약 1.2초 정도 느리지만 이후 확장 데이터 생성 시에는 약 1.6초 정도 빠름을 확인할 수 있었고, 전반적으로 성능이 향상되었음을 확인하였다.

향후 과제로는 본 논문에서 개발된 알고리즘을 NS3에 적용하여 보다 실전적인 실험을 수행하는 것과 실제 네트워크에 적용하는 것이다.

참 고 문 헌(Reference)

[1] R. Enns. Ed, “NETCONF Configuration Protocol”, RFC 4741, Dec. 2006.

[2] Yang, http://www.netconfcentral.org/static/slides/yangtutorial/yang_getting_s24tarted2.html

[3] Mi-Jung Choi et al, ″XML-based Configuration Management for IP Network Devices″, IEEE Communications Magazine, Vol. 41, No. 7, pp. 84-91, 2004. 7.

[4] Ji Huang, Bin Zhang, Guohui Li, Xuesong Gao, Yan Li, ″Challenges to the New Network Management Protocol-NETCONF″, Education Technology and Computer Science, ETCS ’09. First International Workshop on, pp. 832-836, 2009. 6.

[5] Yanan Chang, Debao Xiao, “Design and Implementation of NETCONF-Based Network Management System”, IEEE Computer Society, 2008 IEEE DOI 10.1109/FGCN, pp. 256-259, 2008. 12.

[6] Sun-Mi Yoo, “Performance Improvement Methods for NETCONF-Based Configuration Management”, Computer Science, pp. 242-25.2, 2006.

[7] Huang Ji, "Challenges to the New Network Management Protocol-NETCONF", Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on, pp. 832-836, 2009.
[8] Yanan Chang, Debao Xiao, "Design and Implementation of NETCONF-Based Network Management System", 2008 Second International Conference on Future Generation Communication and Networking, IEEE Computer Society Magazine, pp. 256-259, 2008. 12.1.
[9] Schonwalder J., Bjorklund, M., "Network configuration management using NETCONF and YANG", IEEE communications magazine, Vol.48 No.9, pp.166-173, 2010.
[10] James Yu, Imad Al, "An Empirical Study of the NETCONF Protocol", IEEE Computer Society Magazine, pp. 253-258, 2010. 4.
[11] Myungsook Lee, "Improved Performance of network configuration management and System base on NETCONF", Korea communication science society journal, Vol 33, Issue 9, pp.787-790., 2008.09.
[12] Apostolos E. Nikolaidis et al, "Management Traffic in Emerging Remote Configuration Mechanisms for Residential Gateways and Home Devices," IEEE Communications. Magazine, Vol.43, Issue 5, pp.154-162, May 2005.
[13] Mi-Yang, Cha, Yang-Min Lee, Jae-Kee Lee, "NETCONF Protocol with Improvements Renewal Method for Device Configuration Data", 2013 Korean society for Internet Information Spring Conference Magazine, Vol 14, Issue 1, 2013. 6.
[14] Mi-Yang Cha, Yang-Min Lee, Jae-Kee Lee, "Development Renewal Method for Device Configuration Data of NETCONF Protocol considered Multiple Network Managers", IEEK Summer Conference 2013, Paper CD, 2013. 7.
[15] Mi-Yang Cha, Jun-Suk Won, Jae-Kee Lee, "Network Circuit Auto Recovery using an Improved NETCONF Protocol", 정보과학회논문지 Vol. 18, Issue 9, pp. 628-638 2012. 9.

◐ 저 자 소 개 ◑

이 양 민(Yang-Min Lee)
2000년 동아대학교 컴퓨터공학과(공학사)
2002년 동아대학교 대학원 컴퓨터공학과(공학석사)
2006년 동아대학교 대학원 컴퓨터공학과(공학박사)
2003년~2010년 동아대학교 컴퓨터공학과 강사
2010년~현재 Forhum Software CEO
관심분야 : 유비쿼터스 컴퓨팅, MANET, SDN, NETCONF etc.
E-mail : manson23@nate.com

차 미 양(Mi-Yang Cha)
1997년 동아대학교 컴퓨터공학과(공학사)
2002년 동아대학교 교육대학원 컴퓨터공학과 (컴퓨터교육학석사)
20013년 동아대학교 대학원 컴퓨터공학과(박사과정)
2003년~현재 동아대학교 컴퓨터공학과 강사
관심분야 : NETCONF, MANET, SDN etc.
E-mail : aaa2766@hanmail.net

◐ 저 자 소 개 ◑

이 재 기(Jae-Kee Lee)

1974년 영남대학교 전자공학과(공학사)
1983년 영남대학교 대학원 전자계산학과(공학석사)
1990년 동경대학 공학연구과 전자정보공학과(공학박사)
1984년~1990년 한국전자통신연구원 연구원
1990년~현재 동아대학교 컴퓨터공학과 교수
관심분야 : 유비쿼터스컴퓨팅, SDN, 클라우드컴퓨팅, MANET, 분산모니터링시스템 etc.
E-mail : jklee@dau.ac.kr