http://dx.doi.org/10.7840/kics.2014.39B.6.387

SDN 응용 검증을 위한 프로세스 알지브라 기반 정형 기법

신 명 기[•°], 이 종 화[*], 최 윤 철[*], 이 지 현[*], 이 승 익[*], 강 미 영[**], 곽 희 환[***], 최 진 영[**]

Process Algebra Based Formal Method for SDN Application Verification

Myung-Ki Shin[•°], Jong-Hwa Yi[*], Yunchul Choi[*], Jihyun Lee[*], Seung-Ik Lee[*], Miyoung Kang[**], Hee Hwan Kwak[***], Jin-Young Choi[**]

요 약

최근 SDN (Software Defined Networking) 기반의 다양한 네트워크 제어 및 관리 플랫폼들이 서비스제공자 및 통신사업자들에 의해 연구되고 있다. SDN의 중요한 특징 중 하나는 소프트웨어 프로그램으로 작성된 간단한 응용에 의해 네트워크가 쉽게 제어되고 관리된다는 점에 있다. 이러한 관점에서 잘못 작성된 SDN 응용은 네트워크 전체에 오류를 발생시킬 수 있어, 해당 응용은 작성된 오픈플로우 포워딩 규칙(rule)을 SDN 컨트롤러(controller)를 통해 스위치에 반영하기 전에 토폴로지와 네트워크 환경의 안전성(safety)과 일관성(consistency)이 반드시 검증되어야 한다. 본 논문에서는 SDN 응용 검증을 위한 프로세스 알지브라 (process algebra) 기반의 언어인 pACSR (Packet based Algebra of Communicating Shared Resources)와 이를 기반으로 한 정형 검증 프레임워크를 제안하고, 이에 대한 SDN 정형검증 도구 연구시제품 구현 현황을 기술한다.

Key Words : SDN (Software-Defined networking), OpenFlow, Process Algebra, Formal Method, Verification

ABSTRACT

Recently, there have been continuous efforts and progresses regarding the research on diverse network control and management platforms for SDN (Software Defined Networking). SDN is defined as a new technology to enable service providers/network operators easily to control and manage their networks by writing a simple application program. In SDN, incomplete or malicious programmable entities could cause break-down of underlying networks shared by heterogeneous devices and stake-holders. In this sense, any misunderstanding or diverse interpretations should be completely avoided. This paper proposes a new framework for SDN application verification and a prototype based on the formal method, especially with process algebra called pACSR which is an extended version of Algebra of Communicating Shared Resources (ACSR).

※ 본 연구는 미래부가 지원한 2014년 정보통신·방송(ICT) 연구개발사업의 연구결과로 수행되었음.
•° First Author and Corresponding Author : ETRI (Electronics and Telecommunications Research Institute), mkshin@etri.re.kr, 정회원
* ETRI (Electronics and Telecommunications Research Institute)
** Korea University
*** Solid/Cemware, Co., Ltd.
논문번호 : KICS2014-05-206, Received May 30, 2014; Revised June 16, 2014; Accepted June 16, 2014

Ⅰ. 서 론

최근 SDN 기반의 다양한 네트워크 제어 및 관리 플랫폼들이 서비스제공자 및 통신사업자들에 의해 활발히 논의되고 있다. SDN 기술은 표준 인터페이스 기반으로 데이터 전송 부분으로부터 제어 부분을 분리하고 추상화하여 서비스제공자/통신사업자가 원하는 다양한 네트워크 제어, 관리기능 등을 보다 유연하게 프로그래밍 가능하도록 한다[1,2]. SDN의 중요한 특징 중 하나는 소프트웨어 프로그램으로 작성된 간단한 응용에 의해 네트워크가 쉽게 제어되고 관리된다는 점에 있다. 이러한 관점에서 잘못 작성된 SDN 응용은 네트워크 전체에 오류를 발생시킬 수 있어 응용이 SDN 스위치 등에 규칙을 반영하기 전에 가상적 또는 물리적 토폴로지와 네트워크 환경의 안전성과 일관성이 반드시 검증되어야 한다.

그림 1은 SDN 스위치, SDN 컨트롤러, 응용 프로그램으로 구성된 일반적인 SDN 환경에서 오픈플로우 응용간의 규칙 충돌에 의한 오류 발생 사례를 나타낸 것 것이다. 예를 들어 개발자A 가 작성한 라우팅(routing) 응용과 개발자B가 작성한 방화벽(firewall) 응용은 각각의 독립적 프로그램 상에는 오류가 없더라도 하나의 컨트롤러 상에 다중의 응용이 실행되게 되면 하부 SDN 스위치들에 반영되는 오픈플로우 규칙 간의 충돌 혹은 불일치가 발생할 수 있다.

따라서 본 논문에서는 오픈플로우를 사용하는 일반적인 SDN 환경에서 응용 프로그램의 안전성과 일관성을 사전 검증하기 위한 방법으로 정형 검증 이론을 도입하고, 오픈플로우의 컨트롤러가 유지하는 전역 플로우테이블 동작과 네트워크 토폴로지 정보 등을 정형명세 언어로 모델링하여 이를 심볼릭하게 검증하는 정형 방법론을 제안한다.

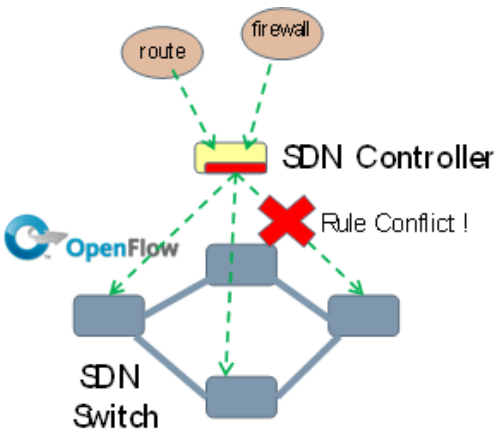


그림 1. SDN 환경에서의 응용간 오픈플로우 규칙 충돌 예
Fig. 1. Example of Rule Conflict between OpenFlow Applications

본 논문의 2장에서는 본 연구에서 새롭게 설계한 프로세스 알지브라 기반의 정형 명세언어인 pACSR와 이를 기반으로 한 SDN 응용 정형 검증 프레임워크를 제안하고, 3장에서는 이를 구현한 연구시제품 동작 예를 소개한다. 4장에서는 본 연구와 직접적으로 비교 가능한 SDN 응용 검증 관련한 이전 연구들에 대해 논의하며, 5장에서는 결론 및 향후 과제 등을 기술한다.

Ⅱ. 프로세스 알지브라 기반 정형검증 프레임워크

본 논문은 SDN 응용의 오류 검증을 위하여 정형 기법 방식을 사용한다. 정형 기법은 수학과 논리학을 기반으로 하드웨어 또는 소프트웨어 시스템을 모호하지 않고 정확히 모델링 할 수 있는 기법이다. 그리고 시스템이 처음에 의도된 요구사항과 동일한지, 더불어 시스템의 안전성과 일관성의 속성들이 보장되는지를 수학적 성질을 이용하여 증명 할 수 있다. 시스템의 요구사항을 엄격히 만족시키고 신뢰성을 보장해야 하는 실시간 임베디드 시스템을 비롯한 다양한 분야에서 주로 활용된다. 정형 의미론 (Formal Semantics)이란 프로그래밍 언어의 의미를 수학 및 논리학 기반으로 엄밀히 정의한 것으로, 언어 및 컴파일러 구현이나 프로그램 검증 및 하드웨어 시스템의 검증 분야에서 주로 활용된다 [3-6]. 본 논문에서는 SDN 정형 검증을 위해 프로세스 알지브라 계열의 명세 언어를 선택하여 이를 확장 설계하였고, 이를 기반으로 SDN을 위한 새로운 정형 검증 프레임워크를 제안한다.

2.1 프로세스 알지브라 기반 SDN 정형명세 언어 - pACSR

ACSR 언어는 미국 펜실베니아 대학에서 개발한 실시간 임베디드 시스템 및 CPS (Cyber Physical Systems) 응용 검증을 위해 설계된 정형명세 언어로서, CCS (Calculus for Communicating Systems)에 기반한 프로세스 알지브라를 이용하여 시간, 자원, 우선순위, 동시성 등 실시간 시스템에 필요한 여러 개념을 포함한다. 실시간 시스템의 정확성은 계산결과가 얻어지는 시간과 밀접한 관련이 있으므로 어떤 사건이 발생하는 시간을 파악할 수 있는 방법을 프로세스 알지브라에서 제공하는 것이 필요하다. 본 연구에서 많은 정형언어 중 프로세스 알지브라 언어 계열인

ACSR를 선택한 이유는 ACSR 자체에는 시간의 개념이 포함되어 있어 시스템을 명확하고 간결하게 묘사할 수 있고 실시간 시스템을 공유자원을 사용하는 상호 통신 프로세스들의 집합체로 정의하여 오픈플로우의 플로우테이블과 같은 자원의 동작을 간결하게 정형 명세 가능하기 때문이다. ACSR 에서 공유자원의 사용은 시간의 경과를 수반하는 액션(action)으로 표현되며, 프로세스 사이의 통신은 동기화된 이벤트(event)로 나타낸다. ACSR의 시간 영역은 이산적(discrete)이거나 조밀(dense)할 수 있으나 본 연구에서는 이산적 이라고 전제했다. 즉, 이벤트가 임의의 시간에 발생하는 것이 아니고 고정된 시간 간격으로 발생 한다고 가정한다. 액션은 시간자원 소모(timed resource consuming) 액션과 즉각적(instantaneous) 이벤트 형태를 지원한다[7-10].

본 논문에서는 기존 ACSR에서 통신 채널을 통해 SDN 네트워크상에 패킷을 주고받을 수 있도록 ACSR을 확장하여 새롭게 정의한 언어를 pACSR라고 명명하였다[11,12]. pACSR 은 기존의 ACSR에 동적 우선 순위 (dynamic priority)와 값-전달 (value-passing) 기법이 추가되었다. 우선순위(priority)는 변수를 포함하는 값 표현 (value expression)으로 나타낼 수 있는데, 예를 들면 시간자원 소모 액션 {(cpu,y+1)} 에서와 같이 y라는 변수로 우선순위를 나타낼 수 있다. 즉각적 이벤트도 값-전달 (value-passing)을 나타내기 위해 값 표현 (value-expression)이 추가되었는데, 예를 들면 (c?x, 3)은 3이라는 우선순위로 c 이라는 채널을 통하여 x라는 값을 받는 입력 동작이고, (c!5, y)는 y라는 우선순위로 c 이라는 채널을 통하여 5라는 값을 내보내는 출력 동작이다. 프로세스는 파라미터(parameter)를 가질 수 있게 되는데, 예를 들면, 두개의 파라미터를 갖는 프로세스 상수 Job은 Job(0,5)를 수행시킴으로써 불려질 수 있다. 여기서 0과 5는 실제 파라미터(actual parameter)로 실제 값으로 공급된다.

본 논문에서 기존 ACSR을 기반으로 확장 설계한 pACSR의 구문은 다음과 같으며 이에 대한 설명은 표 1에 기술한다.

$$
\begin{aligned}
&P ::= NIL \mid A : P \mid e.P \mid be \rightarrow P \mid P_1 + P_2 \mid P_1 \| P_2 \mid \\
&\quad [P]_I \mid P \backslash F \mid P \backslash\backslash I \mid C(\vec{x}) \\
&e ::= (\tau, ve) \mid (l?, ve) \mid (l!, ve) \mid (l?x, ve) \mid (l!ve_1, ve) \\
&A ::= \varnothing \mid S \\
&S ::= (r, ve) \mid (r, ve), S
\end{aligned}
$$

pACSR 프로세스의 실행은 ground term으로 변환되어 레이블이 있는 트랜지션 시스템인 LTS (labeled transition system)로 정의될 수 있다. 예로 프로세스 P1이 다음과 같은 동작을 보일 수 있다.

$$P_1(10.0.1.1) \xrightarrow{a_1} P_2(10.0.1.1) \xrightarrow{a_2} P_3 \xrightarrow{a_3} \cdots$$

"$P(x) \triangleq a_1 P_2(x)$"와 같이 정의되어 있을 경우 프로세스 P1에 10.0.1.1이라는 실제 패킷을 입력 파라미터로 준다. 여기서 10.0.1.1은 source IP를 10.0.1.1로

표 1. pACSR 구문 설명
Table 1. Syntax description of pACSR

pACSR Syntax	Description
NIL	A process that executes no action(i.e., it is deadlocked).
A:P	To execute a timed, resource-consuming action A, consume one time unit, and proceed to the process **P**.
e,P	To execute the instantaneous event e, and proceed to **P**.
P1+P2	A non-deterministic choice between **P**1 and **P**2 which is subject to priority arbitration.
P1||P2	The parallel composition of **P**1 and **P**2 in which the events from **P**1 and **P**2 synchronize or interleave while the timed actions from **P**1 and **P**2, if they do not share any resource, reflect the synchronous passage of time.
[P]I	The close operator, to produce a process that uses the resources in *I* exclusively.
P\F	The restriction operator, to restricts events with labels in **F** from executing.
P\\I	To represent the behavior of **P** in which the identities of resources in I are concealed.
C	A process constant with a certain arity. Each process constant **C** with arity *n* is associated with a *process definition* of the form $C(\vec{x}) \triangleq P$, where $\vec{x}$ is a vector of *n* variables.
be->P	A conditional process term which behaves like P if the boolean expression *be* evaluates to true, or NIL if *be* is false.

가지는 한 패킷의 인스턴스를 말한다. P1(10.0.1.1)이 a1이라는 액션을 실행하고 P2(10.0.1.1) 이라는 프로세스가 되는데, 이 프로세스 P2가 "$P_2(x) \stackrel{\text{def}}{=} foo(x) \rightarrow a_2.P_3$"와 같이 정의되어 있다면 "foo(10.0.1.1)"라는 predicate가 참(true)을 리턴 할 경우 a2는 액션을 실행하고 프로세스 P3가 된다.

또한 pACSR에서는 어떤 액션에 우선순위를 관련시켜서 우선순위가 제공되는데, 예를 들어 a.P + b.Q 에서 a의 우선순위가 b보다 높으면 a가 먼저 실행된다.

다음은 오픈플로우 환경에서 플로우테이블의 동작 연산을 본 논문에서 정의한 pACSR 구문을 이용하여 정형 명세한 예이다.

$$P(x) := matchSrcIP(x, 10.0.1.1) \rightarrow (ch!x, 1).nil$$
$$S := (ch?y, 1).nil$$
$$Sys := (P(10.0.1.1) \parallel S)/ch$$

위 구문의 의미는 P(x)는 packet x를 파라미터로 가지는 프로세스로서, 프로세스 P가 packet x를 받으면 predefined predicate인 "matchSrcIP" 에서 x 의 source IP 가 10.0.1.1 인가를 체크하고, Packet x 의 source IP 가 10.0.1.1 인 것이 확인되면, 프로세스 P는 packet x를 채널 ch 를 통해 보낸다. Packet x를 보낸 후 프로세스 P는 nil이 된다. 만약 packet x의 source IP가 10.0.1.1이 아니어서 matchSrcIP 가 거짓(false)일 경우, 프로세스 P는 nil이 된다. 프로세스 S는 채널 ch를 통하여 packet을 받으며 받은 packet을 y라고 명명한 후 nil 프로세스가 된다. 프로세스 Sys는 프로세스 P(x)와 프로세스 S를 parallel composition하는 프로세스로 P 프로세스의 파라미터로 source IP가 10.0.1.1인 packet을 사용하는 것을 보여주고 있다. 즉, 프로세스 Sys는 프로세스 P(10.0.1.1)과 프로세스 S가 동시에 실행되는 것을 모델링하고 있으며 P(10.0.1.1)의 파라미터인 source IP를 10.0.1.1으로 가지고 있는 packet을 "matchSrcIP" predicate가 참일 경우 채널 ch를 통해 프로세스 S에게 전달하는 것을 보여 주고 있다.

본 논문에서 설계한 pACSR의 중요한 특징 중의 하나는 오픈플로우의 플로우테이블의 동작을 모두 명세하기 위해 Predefined Predicate (PP)와 Predefined Function (PF)라는 함수들을 새롭게 정의하였다는 점에 있다. 앞서 설명하였듯이 "matchSrcIP"는 Predefined Predicate (PP)으로 "matchSrcIP (x,10.0.1.1)" 의 의미는 packet x의 source IP가 10.0.1.1 인가를 체크하는 predicate으로 source IP가 10.0.1.1 일 경우는 참(true)을 리턴 하고 그렇지 않을 경우는 거짓(false)을 리턴하는 기능을 수행한다. Side effect가 없다는 조건하에 여러 다른 Predefined predicates이 정의할 수 있으며, Predefined Function (PF)는 packet을 리턴 하는 함수로 채널 output action(!)이나 프로세스 파라미터로 사용된다.

2.2 pACSR 기반 SDN 응용의 정형검증 방법

본 논문에서 제안한 pACSR 기반의 SDN 정형 검증 프레임워크는 그림 2에서 도식화 한 것과 같이 SDN 컨트롤러, 오픈플로우 정보 저장소 (전역 플로우 테이블, 토폴로지 정보 등), 그리고 pACSR 검증을 위한 veriSDN 코어모듈로 구성된다[12]. 먼저 SDN 컨트롤러에서는 검증 요청 및 결과 수신을 위하여 컨트롤러 내부에 들어가는 veriSDN plug-in 모듈과 기본적으로 실행되는 라우팅 또는 모니터링 응용 프로그램, 그리고 검증을 필요로 하는 응용 프로그램들이 동작한다. 특히 컨트롤러 내부에 들어가는 veriSDN plug-in 모듈에서는 컨트롤러에서 처리하는 다양한 정보의 종류와 특성에 따라 정보 저장소로 전송할 것인지, veriSDN 코어모듈로 전송할 것인지를 분류한다. 그 중 플로우테이블 정보나 토폴로지 정보와 같은 기본정보는 외부 스토리지 모듈에 동기화 하여 저장 하고 플로우테이블의 이벤트 정보, 스위치 이벤트 정보와 같은 다양한 이벤트 정보와 컨트롤러에서 동작하는 응용들과 관련된 정보들은 veriSDN 코어모듈에 전달한다.

검증 프레임워크의 핵심 모듈인 veriSDN 코어 모

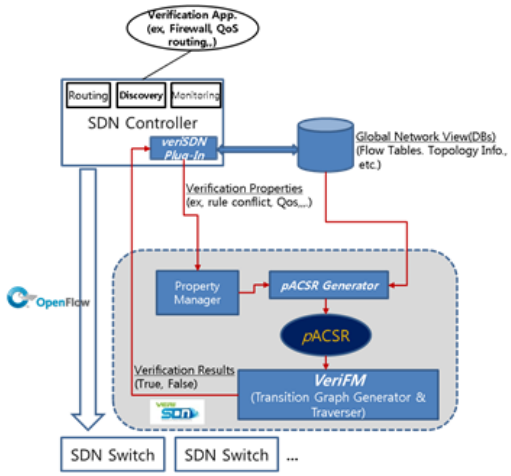


그림 2. pACSR기반 SDN 정형검증 프레임워크
Fig. 2. Framework of SDN Formal Verification based on pACSR

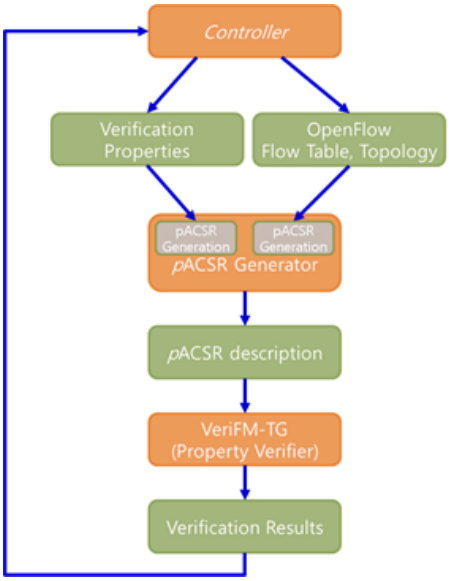


그림 3. SDN 정형검증 동작 순서도
Fig. 3. Operational Process of SDN Formal Verification

툴은 property manager, pACSR generator, pACSR, VeriFM 으로 구성된다. property manager 에서는 SDN 컨트롤러의 veriSDN plug-in 모듈로부터 검증하고자 하는 대상 응용프로그램의 속성 정보를 입력받아 pACSR 정형명세 코드 생성을 위한 전처리 과정을 수행한다. pACSR generator 모듈에서는 외부 스토리지 모듈로부터 토폴로지, 플로우테이블 정보를 읽어 변환규칙에 따라 pACSR 정형명세 코드를 생성하고 여기에 property manager 모듈에서 생성한 검증 속성들도 pACSR 정형명세 코드로 변환하고 이들을 통합하여 pACSR 구문에 부합하는지를 검증하는 과정을 거친다. 이렇게 생성된 pACSR 정형명세 코드를 기반으로 veriFM 모듈에서는 전환 그래프 (Transition Graph)를 생성 한다. 이렇게 생성된 그래프를 기반으로 검증이 필요한 속성에 대하여 입력 플로우 엔트리가 현재의 플로우테이블과 토폴로지 환경에서 오류가 발생하는지의 여부를 확인하게 된다.

그림 3은 정형검증 프레임워크의 동작 순서도이다. 먼저 SDN 컨트롤러에서는 플로우테이블과 토폴로지 정보를 하나의 정보로 간주하고 이와 더불어 검증이 필요한 응용의 검증 속성 정보를 또 다른 하나의 정보로 분리하여 각각 저장하고 관리한다. 이렇게 검증 속성을 따로 관리하는 이유는 SDN 환경에서 다양한 응용들의 개발에 따른 각각의 특성들을 보다 쉽게 적용하기 위함이다. 이러한 정보들은 각각의 pACSR 정형명세 코드로 생성이 되고 pACSR 정형명세시에 pACSR 구문에 맞는지 여부를 확인하는 과정을 거친다. 해당 구문에 부합할 경우에는 VeriFM에서 전환 그래프를 생성하고 여기에서 검증이 필요한 대상 플로우 엔트리가 검증 속성에 대하여 오류가 발생하는지의 여부를 조사하여 해당 결과를 검증을 요청한 곳으로 반환한다. 이 때 검증을 요청하는 모듈에는 SDN 컨트롤러 또는 veriSDN GUI 가 해당된다.

본 논문에서는 SDN 환경에서 정형검증을 위한 pACSR 변환 예제 구성을 위한 SDN 구성도 및 플로우테이블을 그림 4와 같이 설정하고 이에 대한 검증 과정을 단계별로 설명한다. 간단한 예제를 위하여 2개의 스위치(S) 두고 각각의 스위치는 하나의 노드(H)와 연결됨으로써 각 스위치마다 두 개의 포트를 갖도록 설정하였다. 이 때 스위치(S)와 노드(H)의 연결은 채널(CH)로 정의하였다. 각 스위치의 id와 IP 는 다음과 같다. S1은 00:00:00:00:00:11/10.0.2.1 의 정보를, S2는 00:00:00:00:00:22/10.0.2.1 의 정보를 가지며 H1의 mac 정보는 00:00:00:00:11:11, IP는 10.0.1.2로 설정하였다. H2는 각각 00:00:00:00:22:22 와 10.0.2.2로 설정하였다. 이와 함께 rule1은 H1으로부터 들어오는 패킷의 ip 주소를 변경하여 H2에 전달하도록 하고

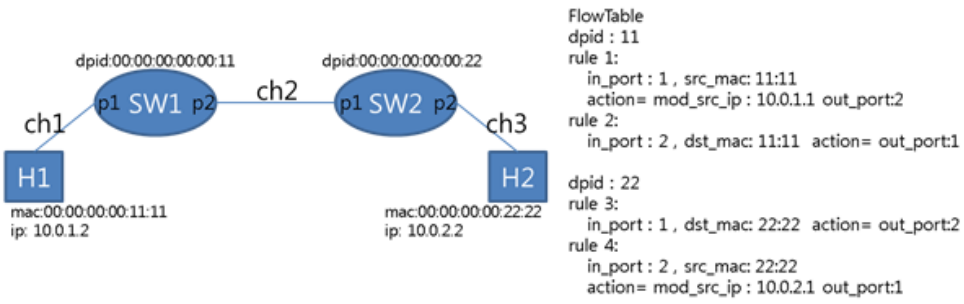


그림 4. Flow table과 토폴로지 뷰 예제
Fig. 4. Example of Flow table and Topological View

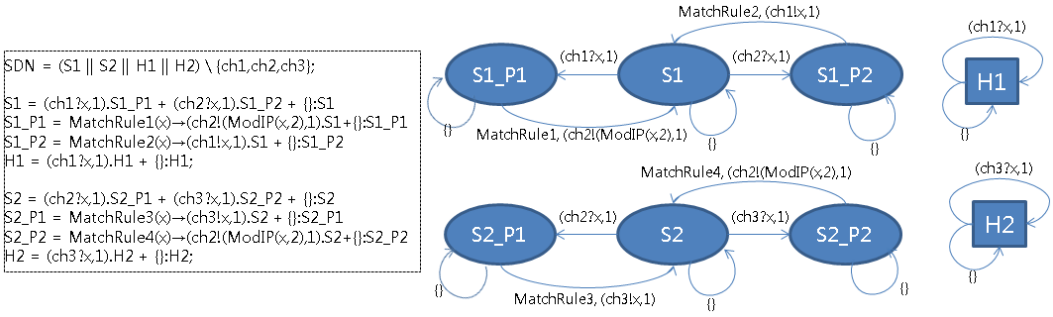


그림 5. pACSR 정형명세에 따른 전환 그래프 생성 예
Fig. 5. Example of pACSR and Transition Graph Generation

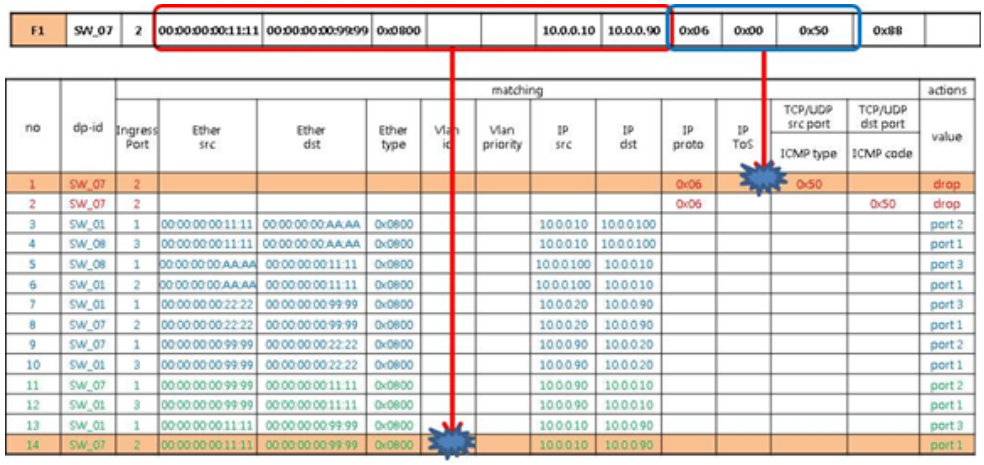
rule4는 H2로부터 전송되는 패킷의 ip를 변경하여 H1으로 전달하도록 설정하고 이는 그림4 에서 확인할 수 있다.

그림 5는 앞에서 설명한 플로우테이블과 토폴로지 정보를 기반으로 pACSR 코드와 전환 그래프를 도식화한 것이다. 그림5 에서 생성된 pACSR 정형명세 코드를 보면 SDN 문은 각각의 스위치(S1, S2)와 노드(H1, H2)를 프로세스로 하고 이에 대하여 병렬 프로세스(||)로 정의하고 채널(ch1,ch2,ch3) 리소스들에 의하여 제한됨(\)을 확인 할 수 있다. 스위치 프로세스(S1, S2)는 포트 프로세스(S1_P1, S1_P2, S2_P1, S2_P2)로 나누어 지고 MatchRule 을 통하여 플로우테이블 매칭과 액션이 수행 됨을 확인할 수 있다. 이렇게 생성된 pACSR 코드로 전환그래프를 생성하게 되면 그림5로 표현이 가능하다.

그림 5의 S1 을 기준으로 ch1에서 패킷이 들어오면 S1_P1 으로 변환이 되고 여기서 입력 플로우 가 플로우테이블과 매칭 되면 IP를 변경하고 S2 의 입력으로 들어가게 되며 이 때 S2 프로세스가 S1 프로세스와 ch2를 통해 동기화 되어 패킷이 전달되게 된다. 즉 S2는 h2로 입력을 받아서 S2_P1으로 이동하고 여기서 해당 플로우가 플로우테이블과 일치하는 경우 ch3으로 내보내고 H2 프로세스를 활성화시킨다. H2 프로세스가 활성화 되어 packet이 오류없이 traverse가 되었으며 모든 프로세스들이 deadlock 없음을 확인 할 수 있다.

Ⅲ. 연구시제품 구현

본 논문에서는 pACSR 정형검증 프레임워크 구현을 위하여 veriSDN 코아 모듈과 mysql DB를 하나의 시스템에 적용하고 시각화를 위한 veriSDN GUI 시스템을 따로 배치하여 2대의 시스템으로 테스트 환경을 구성하였다. veriSDN GUI는 windows 환경에서



F1	SW_07	2	00:00:00:00:11:11	00:00:00:00:99:99	0x0800			10.0.0.10	10.0.0.90	0x06	0x00	0x50	0x88	

no	dp-id	matching												actions
		Ingress Port	Ether src	Ether dst	Ether type	Vlan id	Vlan priority	IP src	IP dst	IP proto	IP ToS	TCP/UDP src port / ICMP type	TCP/UDP dst port / ICMP code	value
1	SW_07	2								0x06		0x50		drop
2	SW_07	2								0x06			0x50	drop
3	SW_01	1	00:00:00:00:11:11	00:00:00:00:AA:AA	0x0800			10.0.0.10	10.0.0.100					port 2
4	SW_08	3	00:00:00:00:11:11	00:00:00:00:AA:AA	0x0800			10.0.0.10	10.0.0.100					port 1
5	SW_08	1	00:00:00:00:AA:AA	00:00:00:00:11:11	0x0800			10.0.0.100	10.0.0.10					port 3
6	SW_01	2	00:00:00:00:AA:AA	00:00:00:00:11:11	0x0800			10.0.0.100	10.0.0.10					port 1
7	SW_01	1	00:00:00:00:22:22	00:00:00:00:99:99	0x0800			10.0.0.20	10.0.0.90					port 3
8	SW_07	2	00:00:00:00:22:22	00:00:00:00:99:99	0x0800			10.0.0.20	10.0.0.90					port 1
9	SW_07	1	00:00:00:00:99:99	00:00:00:00:22:22	0x0800			10.0.0.90	10.0.0.20					port 2
10	SW_01	3	00:00:00:00:99:99	00:00:00:00:22:22	0x0800			10.0.0.90	10.0.0.20					port 1
11	SW_07	1	00:00:00:00:99:99	00:00:00:00:11:11	0x0800			10.0.0.90	10.0.0.10					port 2
12	SW_01	3	00:00:00:00:99:99	00:00:00:00:11:11	0x0800			10.0.0.90	10.0.0.10					port 1
13	SW_01	1	00:00:00:00:11:11	00:00:00:00:99:99	0x0800			10.0.0.10	10.0.0.90					port 3
14	SW_07	2	00:00:00:00:11:11	00:00:00:00:99:99	0x0800			10.0.0.10	10.0.0.90					port 1

그림 6. 규칙 충돌 예가 발생하는 플로우테이블
Fig. 6. Flow Table Example of Rule Conflict

C++ 와 visual studio 2010 sp1을 이용하여 개발하였고, veriSDN core 는 리눅스 fedora 12 버전에서 mysql -server 5.5 를 이용하여 개발 환경 및 테스트 환경을 구축하였다.

pACSR 정형검증 프레임워크의 정확한 동작여부를 판단하기 위하여 규칙 충돌 속성을 검증하였다.이를 위하여 SDN 컨트롤러에 기본적으로 라우팅 응용을 실행시키고 속성 검증을 위하여 보안 응용(TCP 80 port deny)을 추가하였다. 추가된 보안 응용에 따라 그림6의 flow table 1-2가 생성 되고 라우팅 응용에 의하여 flow table 3-13 이 생성 되었다. 이 때 검증을 필요로 하는 flow entry F1이 들어 오고 이에 대한 flow entry 14가 생성되고 오류여부를 검증하였다. 새로 생성된 flow entry 14에 따르면 해당 패킷을 port1으로 전송이 가능하지만 기존 보안 응용에 의해 생성된 flow entry 1에 따르면 패킷을 drop 해야 하므로 규칙 충돌이 발생함을 발견하고 그림 7에서 오류발생 여부를 GUI 를 통하여 확인 할 수 있다.

Ⅳ. 관련 연구

SDN은 네트워크 경로 설정과 제어 및 운용이 소프트웨어적으로 가능하므로 응용 프로그램이 SDN 컨트롤러에 의해 SDN 스위치에 규칙을 반영시키기 전에 안전성과 일관성을 검증하는 것이 필수적이다. 이를 검증하기 위해 활용되는 여러 방법론 중의 하나로 정형기법을 사용하는데, 정형 기법은 SDN의 컨트롤러나 스위치의 네트워킹 기능과 동작을 수학적으로 모델링하고, 수학적 분석을 수행하여 SDN 전체 시스템에 대한 신뢰성과 안정성을 확보할 수 있다는 기대에서 많이 활용되고 있다.

VeriFlow[13]는 초기 제안된 대표적인 SDN 응용 검증 도구로서, 네트워크 상태가 지속적으로 변화하는 과정에서 네트워크 성능엔 영향을 미치지 않으면서 실시간적으로 네트워크 전반의 불변성(invariants)를 검증하기 위해 설계된 방식이다. VeriFlow는 SDN 컨트롤러와 스위치들 사이에 위치하며, 컨트롤러에 전달되는 각 포워딩 규칙 (예: 삽입, 변경 또는 삭제)에 대해 네트워크 전체의 불변성에 위반되는지를 체크한다. 이를 위해 모든 포워딩 상태 변화들을 추적하여 모든 규칙을 취합하고, 네트워크에 도착하기 이전에 가져온 규칙들을 검증하는 방식을 사용한다. VeriFlow에 대한 프로토타입은 NOX 컨트롤러와 미니넷 오픈플로우 네트워크를 이용하여 구현되었으며, 새로운 규칙 삽입 또는 삭제의 경우 백만분의 일초안에 검증이 수

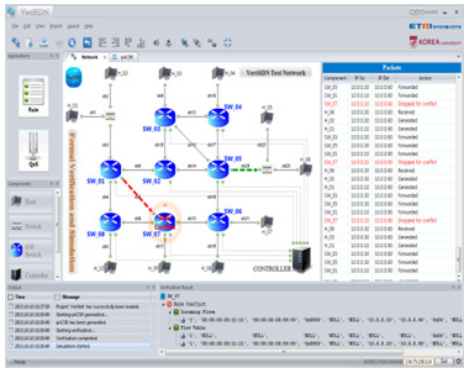


그림 7. 연구시제품 구현결과 화면
Fig. 7. Snapshot of Implementation of Prototype

행됨을 보여준다. 다만 VeriFlow의 경우는 정형 검증이 아닌 단순 포워딩 그래프를 이용하여 검증 하기 때문에 응용이 가지는 논리적 오류 혹은 네트워크 환경의 잠정적 보안 오류와 같은 속성들을 검증하기에는 어렵다.

NICE[14]는 단일 SDN 컨트롤러와 다수의 스위치가 분산되어 있는 확장성을 요구하는 SDN 환경에서의 적합한 정형검증 도구를 제공한다. NICE는 입력 정보로써 컨트롤러에 삽입되는 응용, 네트워크 토폴로지 및 검증을 하고자 하는 속성 정보를 받아 내부적으로 시스템 상태도를 구성하고, 스위치로부터 전달된 패킷에 대해 상태 전이에 따라 검증하는 방식을 사용하고 있다. NICE가 고려한 검증 속성으로는 루프 발생, 패킷이 전달되지 못하는 블랙 홀, 패킷이 수신지에 정확히 도달하는지, 두 호스트간에 양 방향으로 정확히 패킷이 전달되는지, 스위치에서 버퍼에 쌓여 있는 모든 패킷을 다 처리했는 지에 대한 여부를 검증 대상으로 하고 있다. NICE는 실시간으로 변하는 네트워크 환경이나 컨트롤러 상에 동작하는 다중 응용간의 호환성을 검증하는 측면에서는 제약이 있다.

Formal Network Testing[15]은 정형기법을 이용하여 네트워크에서 발생하는 실시간적 오류들을 발견할 수 있는 접근 방식을 제안한다. 각 스위치에 쌓여있는 모든 규칙 정보를 수집하여 전체 네트워크 토폴로지 정보로 활용하고, 각 스위치에서 규칙을 수행하도록 하고, 주기적으로 시험 패킷을 보내 에러가 발생하는 경로나 위치를 찾아 내는 방법을 활용하였다. 검증 속성으로는 패킷에 대한 포워딩 규칙 오류, 링크나 포트 상의 오류를 찾아 내는 경우를 프로토타입으로 구현하였다. 본 방법은 Header Space Analysis[19] 라이브러리를 이용하여 테스트 패킷을 생성하기 때문에 Header Space Analysis 라이브러리의 제한적인 속성

에 대해서만 검증이 가능하고 실시간 적인 통계정보를 이용한 QoS 라우팅이나, 다중패스 라우팅 같은 속성 검증을 요구하는 환경에서는 적용이 어렵다.

Frenetic[16,17]은 기존의 오픈플로우와 NOX API를 이용하는 경우의 한계점 (예를 들면, 하나 이상의 이벤트 처리를 위한 조합(composition) 기능, 하드웨어 스위치로 인한 프로그램 인터페이스 복잡성 등)을 해결하기 위해 만들어진 상위 레벨 프로그래밍 언어이다. 이를 위해 쉽게 추상화하여 패킷을 기술할 수 있는 통합된 구조와 알지브라 기반의 패턴을 제공해 주며, 여러 규칙이 조합되어 처리될 수 있는 방식을 정의한다. 또한, Frenetic 은 오픈플로우 기반의 규칙들을 정형적으로 명시할 수 있는 네트워크 프로그래밍 언어도 제공한다. Frenetic은 상위 레벨 프로그램 언어를 제공 하기 때문에 동적으로 변하는 네트워크 환경에서 나타날 수 있는 문제에 대하여 검증 하기에는 제약이 따른다.

FlowChecker[18]는 여러 오픈 플로우 인프라 도메인에 존재하는 스위치들간 발생할 수 있는 규칙의 오류를 해결하기 위한 검증 구조와 방식을 제안한다. 이를 위해 동일한 도메인에서는 컨트롤러와 여러 스위치간 Flowvisor를 두어 하나의 컨트롤러내의 플로우를 제어하고, 다수 컨트롤러와 스위치들로 구성된 여러 도메인간은 FlowChecker 모듈이 연결되는 구조를 사용하였다. FlowChecker는 각 컨트롤러로부터 플로우 테이블 정보를 수집 및 조합하여 전체 네트워크에 대한 플로우 테이블의 정확성과 일관성을 검증하는 방식이다. 여러 도메인간의 플로우 정보를 검증할 수 있는 정형 모델링 구조와 방식을 정의한다. FlowChecker는 네트워크 환경에 대한 검증은 가능하나 응용들 간에 발생 가능한 리소스 분배 문제나 보안 응용과 같이 특정 응용에서 발생 할 수 있는 문제를 검증 하기에는 적합하지 않다.

Header Space Analysis[19]는 네트워크상의 포워딩 에러 및 구성 에러를 체크할 수 있는 정형 검증 프레임워크를 정의한다. 이 검증 프레임워크는 패킷 수신 에러와 포워딩 루프의 발생여부를 검증하며, 호스트, 사용자 또는 트래픽 격리 시 오류 발생여부를 체크할 수 있는 프로토콜 독립적인 검증 정형 기법을 정의한다. 그러나 대부분 현재까지 제안된 이전 연구와 마찬가지로 이 방식도 지속적으로 변화되는 플로우테이블의 정보를 실시간으로 반영시키지는 못하는 한계를 가지고 있다.

V. 결론 및 향후 과제

본 논문에서 제안한 SDN 응용 검증 방법에서는 프로세스 알지브라 언어 기반의 pACSR을 이용하여 오픈플로우의 플로우테이블의 동작 및 토폴로지 정보 등을 정형 명세하고 이를 기반으로 SDN 응용에서 발생할 수 있는 오픈플로우 규칙간의 충돌 등의 오류를 사전에 방지하고 검증할 수 있음을 보였다. 이전의 다른 연구와 본 연구의 차이점은 프로세스 알지브라 언어인 pACSR 명세 언어가 오픈플로우 플로우테이블과 토폴로지 정보 등으로부터 내부적으로 자동 변환 생성되어 기존의 C, 자바, 파이손 등을 통해 프로그래밍 하는 SDN 응용 개발자는 복잡한 정형명세 언어를 사전에 이해하고 이를 통해 직접 정형 명세 언어로 프로그래밍 할 필요가 없다는 점에 있다. 또한 정형명세 언어로서 기존 프로세스 알지브라 계열의 ACSR 를 확장하여 SDN의 자원인 플로우테이블의 동작 등을 실시간 시스템내 공유자원을 사용하는 상호 통신 프로세스들의 집합체로 정의하는 등, 시간의 개념이 포함된 SDN 자원의 동작을 보다 간결하게 명확하게 정형 명세 가능하다.

본 연구는 아직 연구단계로서 실용단계로 발전하기 위해서는 아래와 같은 몇 가지의 이슈가 사전에 해결되어야 한다. 먼저 정형 검증을 위해 사용되는 전환 그래프는 플로우테이블과 해당 스위치 수가 늘어나게 되면 그 상태가 증가하여 확장성(scalability)에 문제를 갖게 된다. 이러한 문제를 해결하기 위해 전환 그래프의 검증 적용 시에 상태가 변화된 부분만 적용시키는 점진적 검증 방법 (incremental method)에 대한 추가 연구가 필요하며, 동시에 pACSR와 전환 그래프를 여러개의 CPU를 갖는 멀티코어 프로세서 컴퓨터를 이용하여 분할 계산 및 병렬 처리 될 수 있도록 하는 성능 개선에 대한 연구도 향후 중요한 과제 중 하나이다. 추가적으로 사용자가 임의로 정의한 속성을 동적으로 추가하여 검증하기 위한 방법으로 기존 CTL (Computation Tree Logic) 과 같은 속성 정의 언어를 본 검증 프레임워크에 추가하는 방안도 본 도구의 상용화 연구를 위해 장기적으로 고려 중에 있다.

References

[1] ONF (Open Networking Foundation), OpenFlow Switch Specification Version 1.3, September, 2012, from https://www.opennetworking.org/.

[2] N. McKeown, T. Anderson, H. Balakrishnan,

G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: Enabling innovation in campus networks,” *SIGCOMM CCR*, vol. 38, no. 2, pp. 69-74, 2008.

[3] E. Clarke and J. Wing, “Formal methods: State of the art and future directions,” *ACM Computing Surveys*, vol. 28, no. 6, pp. 626-643, Dec. 1996.

[4] D. Park, “Concurrency and automata on infinite sequences,” in *Proc. GI-Conf. Theoretical Comput. Sci.,* vol. 104, pp. 167-183, *London, UK*, Mar. 1981.

[5] R. Milner, *Communication and concurrency*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[6] E. M. Clarke, E. A. Emerson, and A. P. Sistla, “Automatic verification of finite-state concurrent systems using temporal logic specifications,” *ACM Trans. Programming Languages and Systems*, vol. 8, pp. 244-263, 1986.

[7] H. Kwak, J. Choi, I. Lee, and A. Philippou, “Symbolic weak bisimulation for value-passing calculi,” Technical Report, MS-CIS-98-22, Dept. Comput. Inf. Sci., Univ. of Pennsylvania, 1998.

[8] H. Kwak, I. Lee, and A. Philippou, and J. Choi, “Symbolic Schedulability Analysis of Real-time Systems,” *IEEE Real-Time Systems Symp.*, Dec., 1998.

[9] I. Lee, A. Philippou, and O. Sokolsky, “Resources in process algebra,” *J. Logic and Algebraic Programming*, vol. 72, pp. 98-122, 2007.

[10] J. Choi, I. Lee, and H. Xie, “The specification and schedulability analysis of real-time systems using ACSR,” in *Proc. IEEE Real-Time Systems Symp.*, Dec. 1995.

[11] M. Kang, E. Kang, D. Hwang, B. Kim, K. Nam, M. Shin, and J. Choi, “Formal modelling and verification of SDN-OpenFlow,” *IEEE Conf. ICST*, pp. 481-482, Luxembourg, Luxembourg, Mar. 2013.

[12] M-K. Shin, H. Kwak, J. Choi, and M. Kang, “VeriSDN: Formal verification for software-defined networking (SDN),” *Telecommunication Rev.*, 2013.

[13] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, “VeriFlow: Verifying network-wide invariants in real time,” HotSDN’12, Helsinki, Finland, Aug. 2012.

[14] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, “A NICE way to test OpenFlow applications,” *NSDI,* Apr. 2012.

[15] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, “Formal network testing,” 2012.

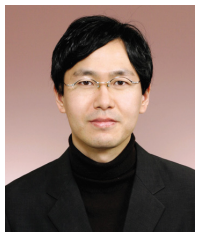
[16] N. Foster, R. Harrison, M. L. Meola, M. J. Freedman, J. Rexford, and D. Walker, “Frenetic: A high-level language for OpenFlow networks,” in *Proc. PRESTO’10*, 2011.

[17] Frenetic project, from http://frenetic-lang.org/index.php.

[18] E. Al-Shaer and S. Al-Haj, “FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures,” in *Proc. SafeConfig’10*, pp. 37-44, Oct. 2010.

[19] P. Kazemian, G. Varghese, and N. McKeown, “Header space analysis: Static checking for networks,” *USENIX Symp. Networked Systems Design and Implementation (NSDI ’12)*, Apr. 2012.

신 명 기 (Myung-Ki Shin)



1992년 2월 : 홍익대학교 전자계산학과 학사

1994년 2월 : 홍익대학교 대학원 전자계산학과 석사

2003년 8월 : 충남대학교 대학원 컴퓨터공학과 박사

1994년 2월~현재 : 한국전자통신연구원 표준연구센터 실장/책임연구원

<관심분야> SDN, NFV, 미래인터넷, IoT

이 종 화 (Jong-Hwa Yi)

1990년 2월 : 한양대학교 전자공학 석사
1996년 12월: 마드리드 국립대학교 정보통신학과 박사
1990년2월~현재 : 한국전자통신연구원 표준연구센터 네트워크표준연구실 근무
<관심분야> SDN, NFV, 네트워킹 검증기법, 미래 인터넷 등

최 윤 철 (Yunchul Choi)

2007년 8월 : 충남대학교 전자정보통신공학과 학사
2010년 2월 : 충남대학교 전자정보통신공학과 석사
2012년 11월~현재 : 한국전자통신연구원 연구원
<관심분야> SDN, OpenFlow

이 지 현 (Jihyun Lee)

2005년 2월 : 경북대학교 정보통신학과 석사
2009년 2월 : 경북대학교 컴퓨터과학과 박사
2009년 5월~현재 : 한국전자통신연구원 네트워크표준연구실 선임연구원
<관심분야> SDN, NFV, 무선이동통신, IoT

이 승 익 (Seung-Ik Lee)

2000년 2월 : 한동대학교 전산전자공학부 졸업
2002년 2월 : 한국과학기술원 정보통신공학 석사
2009년 2월: 한국과학기술원 정보통신공학 박사
2009년 5월~현재 : 한국전자통신연구원 선임연구원
<관심분야> SDN, NFV, NGSON

강 미 영 (Miyoung Kang)

1999년 2월 : 동국대학교 전산학과 석사
2011년 8월 : 고려대학교 컴퓨터학과 박사
2014년 3월~현재 : 고려대학교 고품질융합SW센터 연구교수
<관심분야> 소프트웨어공학, 정형기법, 프로토콜검증, SDN

곽 희 환 (Hee Hwan Kwak)

1986년 : 남일리노이대 컴퓨터학과 학사
1988년 : 캘리포니아 주립대 컴퓨터학과 석사
2000년 : 펜실베니아 대학교 (Upenn) 컴퓨터학 박사
2010년 9월~현재 : 쏠리드 연구위원
<관심분야> 정형기법, SDN, 4G 무선통신, IoT

최 진 영 (Jin-Young Choi)

1982년 2월 : 서울대학교 컴퓨터공학과 졸업
1993년 : 펜실베니아 대학교 (Upenn) 컴퓨터학 박사
1996년 3월~현재 : 고려대학교 교수
<관심분야> 정형기법, 시큐어 소프트웨어 공학, 소프트웨어 보증, SDN