오픈소스 시대의 R&D, 전주기적 활동과 동시공학적 접근

함진호*, 남기혁**, 김형준*
한국전자통신연구원*, 프리스티**

요 약

소프트웨어에 이어 하드웨어에서도 오픈소스시대가 열리고 있다. 소프트웨어와 하드웨어 분야에서의 오픈소스의 폭넓은 확산은 ICT 분야의 R&D 활동에서 새로운 연구 방법과 연구 분야의 확장을 가능케 할 것으로 생각된다.

오픈소스 생태계 하에서는 개별 연구조직의 크기나 연구 규모에 따른 장점은 줄어드는 반면 개인의 창의력이나 협업 능력이 연구경쟁력의 원천이 된다. 이에 따라 과거에는 시도하지 않았던 창의적인 선행연구나 시스템적인 융합연구도 가능해질 것이다.

R&D는 단지 구현결과물을 얻는 것에 한정되는 것이 아니라, 개념 도출, 이에 따른 설계, 소프트웨어 및 하드웨어 구현, 적합성 및 상호운용성시험, 대규모 실증사업까지의 단계적이고 체계적인 추진과 이에 대한 문서화, 규격화 및 표준화 작업들을 포함한다. 이러한 작업은 순차적이 아니라 동시적으로 수행될 때 상호연계 검증에 따른 완성도 제고와 전주기 활동에 있어 총 시간 단축을 꾀할 수 있다.

본 고에서는 오픈소스 생태계에서 연구자들이 이러한 활동의 효과적인 수행과 협력적 융합연구를 위한 저비용 오픈소스 R&D 동시개발 플랫폼 방안을 제시한다.

I. 오픈소스 환경

1. 오픈소스 소프트웨어

오픈소스를 떠나 R&D를 이야기할 수 없는 시대가 되어가고 있다. 아주 독창적인 개념이나 기술이 아닌 한, 홀로 연구를 수행한다는 것은 실제로 되는지 어떤지를 스스로 한번 살펴보았다라는 것 이상의 의미를 가질 수 없게 되었다. 그렇게 도출된 연구결과는 다른 사람들이 사용하기 어렵고 따라서 아무도 사용하려하지 않을 것이기 때문이다.

폐쇄적인 연구가 의미있는 경우는 강력한 방식특허로 연구결과를 보호받을 수 있을 때이다. 소프트웨어와 같이 생명주기가 길고, 성공 시 지속적인 유지보수가 필요한 분야에서는 개방적인 연구가 필수적이다.

오픈소스 생태계에서는 커뮤니티 발전에 얼마나 기여하고 있는가에 따라 영향력을 갖게 되며, 이는 참여자 개개인의 능력과 수고에 따라 차별화된다. 오픈소스 생태계에서는 개개인의 역량이 매우 중요한데, 만일 이들의 능력이 각자의 커뮤니티의 영역을 넘어 다양한 커뮤니티의 전문가들과 긴밀히 협력할 수 있는 환경이 만들어 진다면 기술의 발전은 급물살을 탈 수 있을 것이다.

2. 오픈소스 하드웨어

소프트웨어 분야에서의 오픈소스화는 역사가 30년이 넘는 반면에 하드웨어가 오픈소스화 되기 시작한 것은 비교적 최근의 일이다[1].

Open Source Hardware Association (OSHWA)[2][3]에 의하면 오픈소스 하드웨어란 누구나 제시된 디자인의 하드웨어를 배우고, 수정하고, 배포하고, 제조하고 팔 수 있도록 공개된 하드웨어이다. 하드웨어를 만들기 위한 디자인 소스는 그것을 수정하기에 적합한 형태로 전달될 수 있어야 하고, 오픈소스 하드웨어는 각 개인들이 하드웨어를 만들고 이 하드웨어의 사용을 극대화하기 위하여, 쉽게 구할 수 있는 부품과 재료, 표준 가공방법, 개방된 시설, 제약이 없는 콘텐트 그리고 오픈소스 디자인 툴을 사용하는 것이 이상적이다. 오픈 소스 하드웨어는 디자인을 자유롭게 교환함으로써 지식을 공유하고 상용화를 장려하여 사람들이 자유롭게 기술을 제어할 수 있도록 한다.

오픈하드웨어를 이용함에 따라 당장에 필요하지 않은 하드웨어 개발을 건너뛰어 전체 개발시간을 단축하면서, 바로 소프트웨어 개발로 들어갈 수 있다[4].

하드웨어는 시간이 감에 따라 성능이 지속적으로 향상되는 속성이 있다. 미리 하드웨어를 개발을 해 놓으면 곧 경쟁력을 잃어가기 때문에 사용하려는 시점에 하드웨어를 제작하는 것이

유리하다.

오픈 하드웨어 플랫폼의 성능은 지속적으로 향상되고 보다 다양해지고 있다. Kickstarter.com을 통해 개발된 ONetSwitch[5]는 단지 300불 정도의 저렴한 하드웨어 플랫폼에 오픈플로우를 탑재하고, 무선랜을 원하는 바에 따라 구현하고, SDN을 실현할 수 있다. 이런 플랫폼에서 검증된 소프트웨어적인 기능은 실제로 시스템을 사용하고자 할 때 원하는 성능의 하드웨어 시스템을 제작하여 OS를 맞추어 설치하고 소프트웨어를 이식하면 된다.

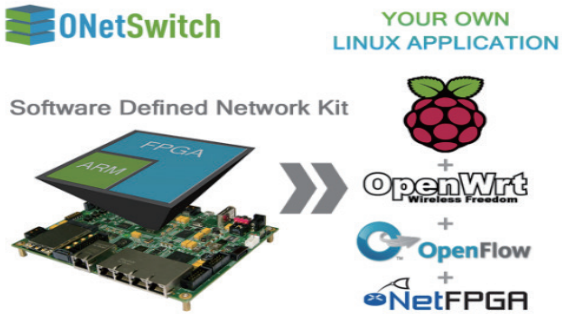


그림 1. SDN을 위한 오픈소스 하드웨어

이러한 하드웨어 플랫폼의 성능(performance)은 최종적으로 기대하는 제품 수준에는 수십분의 일에 불과할지 모르지만 탑재 소프트웨어의 기능(function)은 그대로 실현해 볼 수 있어 개발 플랫폼으로서는 매우 유용하다.

이러한 플랫폼은 가격이 매우 저렴하여 이들 플랫폼이 최종적으로 탑재될 인프라의 모의형상까지를 자체적으로 쉽게 구축해 볼 수 있기 때문에 개발자들이 기능 구현에서부터 개별시스템 시험, 인프라 설치 상황에서의 상호운용성 시험까지의 폭넓은 연구를 수행하는 것이 가능하다. 또한 개별단위의 테스트베드를 서로 연결하여 검증해 볼 수 있다면 국가 단위의 인프라 형상까지도 축소하여 대규모 실험을 운영해 보는 것이 가능하다.

국내에선 경희대 이성원교수연구실과 광주과기원 김종원교수연구실을 중심으로 오픈소스 하드웨어 기반의 플랫폼 및 인프라 연구가 활발히 진행되고 있다.

이성원교수가 발표한 "오픈소스 소프트웨어 관점 기반 SDI 발전 고찰" 논문[6]에서는 SDI(Software Defined Infrastructure)라는 개념에서 SDN, NFV, Cloud Computing을 유기적으로 통합하고, 이를 5G 이동통신까지 확장하는 개념을 제시하고 있다.

김종원교수는 Converged Software-Defined Infrastructure[7]로서 SDN/NFV/Cloud를 통합하는 방안을 제시하고, SmartX Box라는 개념을 통해 다양한 실험을 해볼 수 있는 샌드박스 개념을 플랫폼을 제시하고 있다.

Ⅱ. 연구 개발

연구개발이란 사용자가 요구하는 기능에 대하여 적은 비용으로 빠른 시간에 구현결과를 얻고자 하는 작업이다. 하지만 연구개발은 단지 구현결과물을 얻는 것에만 국한되지 않고, 연구결과물의 완성도를 높이기 위한 다양한 작업들로 구성된다.

1. 기존의 연구개발

구현된 결과물을 반복적으로 재현하기 위해서는 규격이 필요하다 이러한 규격화작업이 특정 개발에 의존적이 되지 않도록 하기 위해서는 시스템을 추상화한 표준화작업이 진행된다. 표준화를 통해서 얻고자 하는 장점은 보다 많은 개발자가 해당결과물을 독자적으로 구현하고, 이들 간에 호환성을 부여할 수 있어 시스템 구축을 보다 쉽게 하고 규모의 경제를 이루기 위한 것이다.

구현결과물이 제대로 구현되었는지를 검증하기 위해서는 적합성시험이 이루어져야 하고, 이를 위해서는 시험규격이 만들어져야 한다. 시험규격은 구현규격에서 제시한 바를 검증하기 위하여 각 항목에 대하여 대응하여 작성된다. 시험규격에 따라 시험기나 테스트 케이스가 개발되면 시험이 이루어진다.

시험이 특정 개발품의 관점에서 벗어나 좀 더 범용성을 갖고 이루어지기 위해서는 관련자들이 함께 모여 시험표준을 만들기도 한다. 이러한 관점에서 구현결과물과 시험기/테스트 케이스, 구현규격과 시험규격, 기능표준과 시험표준은 상보적 관계에 있다.

적합성시험은 단위 시스템의 범주에서 당초 설계에 따라 제대로 구현되었는지를 검증하는 작업이다. 하지만 시스템이 제대로 구현되어있다 할지라도 인프라에 탑재되어 다른 시스템과 연동되면서 문제가 나타날 수 있다. 이러한 문제점을 사전에 해결하기 위해서는 단대단 상호운용성시험이 이루어져야 한다.

이를 위해서 상호운용성규격에 따라 테스트 시나리오가 작성되어야 하며 테스트케이스에 따라 상호운용성시험이 이루어져야 한다.

앞에서 언급한 바와 같이 연구개발은 단지 구현결과만을 얻는 작업이 아니라, 일련의 규격과 시험도구, 시험환경까지를 모두 확보하려는 작업이다. 이들 작업들의 기술적인 사항은 서로 밀접하게 연관되어 있으므로 개발과정에서의 크로스체크를 통해

시스템에 대한 이해도와 완성도를 높일 수 있다.

지금까지는 이러한 작업들이 대부분 시차를 갖고 순차적으로 이루어져 왔다. 이에 따라 각 단계에서 시간이 누적되고 초기 설계에서부터 인프라에 탑재되어 운영될 수 있는 수준의 결과물을 얻는 최종단계에 이르기까지 오랜 시간이 소요되게 된다.

순차적인 개발에서의 또 다른 문제점은 내재된 오류가 즉시 발견되지 못하고 오랜 시간이 지난 후에 발견된다는 점이다.

2. 구현과 시험의 동시공학적인 접근

동시공학(concurrent engineering)은 90년대 말, 2000년대 초에 생산(manufacturing) 공정에서 도입된 산업공학적인 방법으로 순차적으로 진행되던 설계, 구현, 테스팅 과정을 동시적으로 반복 수행함으로써 총 소요시간 단축과 완성도를 높이고자 하는 방법이다. 본 고에서는 이러한 문제해결 방식을 오픈소스 하드웨어, 소프트웨어를 기반으로 하는 ICT 기반의 R&D에 적용하는 방안에 대하여 살펴본다.

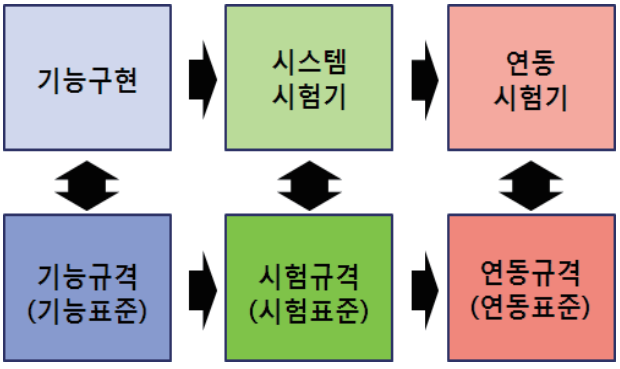


그림 2. 연구개발의 영역과 관련성

그림에서와 같이 개념설계를 통해 기능규격이 작성되고 이에 따라 기능구현이 이루어지는 데, 이때 구현 측면과 시험측면에서 동시적인 검토가 이루어지면서 구현규격과 시험규격을 서로 대응하여 작성된다. 이때 구현은 시험을 염두에 두고, 시험은 구현을 염두에 두고 개발이 진행되면서 상보적인 작업이 이루어질 수 있다. 이에 따라 규격을 작성하는 과정에서 구현과 시험 간에 1차적인 크로스체크가 일어난다. 이는 일종의 사고실험(thought experiment) 과정으로 문제를 비판적 관점(critical thinking)에서 바라보면서 진행하는 것이다. 이러한 과정을 통해서 규격에 대한 골격이 만들어지게 된다.

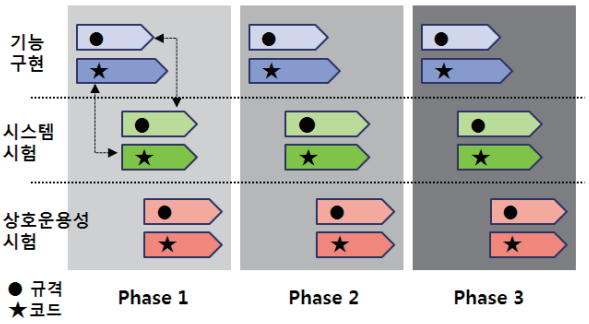


그림 3. 동시공학적 단계별 연구

규격이 개발되면 규격에 따라 구현물과 시험기를 개발하여 이들 간을 연결하여 시험을 수행한다. 이를 통해 2차적인 크로스체크가 일어난다. 1차 크로스체크에서 미처 걸러지지 못한 설계상의 오류들은 2차 크로스체크 과정을 통해서 발견되어 수정될 수 있다.

이러한 과정은 시스템 설계에 큰 영향을 미치는 부분부터 시작해서 보다 세밀한 부분으로 들어가면서 설계를 하게 된다. 일종의 나선형(spiral) 개발방법론이고 애자일 프로그래밍이며, 관점을 달리하는 짝 프로그래밍이다. Phase가 진행하면서 완성도가 향상된다.

이러한 과정은 동시적으로 일어나는 데, 동시공학적인 방법을 사용하면 설계나 구현 이외의 오류 발생의 가능성이 낮아지며 완성도 높은 시스템은 획득하는 시간이 단축된다.

3. 상호운용성시험

상호운용성시험을 수행하기 위해서는 많은 시스템이 관여하여야 한다. 지금까지는 많은 시스템 자원을 독점하여 혼자 상호운용성시험을 하는 것은 쉽지 않았지만 오픈소스 하드웨어 환경 하에서는 가능하다.

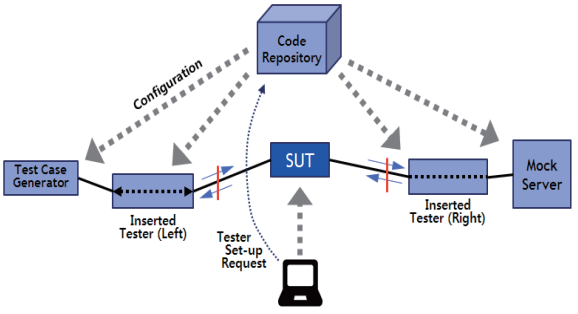


그림 4. 상호운용성시험을 위한 환경

여기에서 상호운용성시험을 수행하기 위한 방법론을 실현해 볼 수 있는 환경에 대하여 제시한다.

위의 그림에서 SUT (System Under Test)는 적합성시험을 통해서 단위 시스템의 완성도가 검증된 시스템이다. SUT에 대한 상호운용성시험을 수행하기 위해서는 양쪽에 있는 연결시스템의 환경을 구성하여야 하고, 상황에 따라 발생한 패킷을 SUT로 보내고, 이를 SUT에서 수행되어 이웃한 시스템으로 전달하고, 리턴되어 온 패킷을 다시 받아들여 처리한 다음 다시 이웃한 시스템으로 넘겨주어야 한다.

상호운용성시험을 원활히 수행하기 위해서는 이러한 환경을 쉽게 구축할 수 있어야 하고, 패킷이 전달되는 상황에서 각각의 패킷에 대한 취득 및 저장, 분석 등의 작업들이 필요하게 된다.

그림에서 SUT의 왼쪽 끝에 있는 시스템은 Test Case Generator이고, 오른쪽 끝에 있는 시스템은 전달되어온 패킷에 반응하는 서버이다. SUT에 각기 이웃한 시스템은 Inserted Tester(IT)로서 지나가는 패킷을 transparent하게 캡춰하고 향후 분석할 수 있도록 지원하는 시스템이다.

이러한 상호운용성을 위한 환경은 SUT에 의존적으로 마련되어야 하므로 각 경우가 맞춤형으로 진행되어야 하기 때문에 이러한 환경구축은 시험자 단말에서의 환경설정 요청에 따라 오픈하드웨어 기반의 각 bare machine을 configuration 함으로써 이루어질 수 있다.

이후 시험자는 Test Case Generator에 테스트케이스를 생성하도록 명령을 내리고, SUT에 전달되는 패킷을 IT(Left) 시스템에서 캡처하고, SUT에서 나가는 패킷에 대하여 IT(Right) 시스템에서 캡처함으로써 실시간 또는 로그된 파일을 분석하는 상호운용성시험 결과 검증 작업을 수행할 수 있다.

그림은 상호운용성시험 간략도로서 실제로는 보다 많은 Test Case Generator와 Mock Server, Inserted Tester가 시험에 참여할 수 있다. 따라서, 상호운용성시험 환경이 갖춰지기 위해서는 방대한 환경 설정 전처리 작업을 해야하는 오버헤드를 부담해야 한다.

이런 오버헤드를 최소화하기 위해서 도커와 쿠버네티스기반를 기반으로 여러 시스템에 대하여 커스터마이즈된 환경을 동시에, 반복적으로 설정할 수 있는 오픈하드웨어 기반의 R&D 플랫폼을 구축한다.

오픈하드웨어 R&D 플랫폼의 역할은 기능구현, 실험, 표준개발 등 R&D 활동의 동시적 수행을 용이하게 하는 개념검증용 플랫폼으로서, 리눅스 컨테이너 기반의 네트워킹 및 서버 컴퓨팅 통합 환경을 구축하고, 통합 환경하에서 다양한 네트워킹 (SDN/NFV, OpenFLow, P4, IoT, ...) 과 컴퓨팅 (AI, Big Data, 분산처리, Simulation, ...) 기능을 여러 개발자가 동시에 분산, 협업 형태로 개발할 수 있는 자원과 시험환경을 제공해 개방형 연구협업을 구축할 수 있도록 지원하는 시스템을 말한다.

III. 오픈하드웨어 기반의 R&D 동시개발 플랫폼

최근 데이터센터 및 클라우드 환경에서 컨테이너 기술을 적극 도입하고 있다. 따라서 현재 상당 수의 소프트웨어 배포 환경이 도커와 같은 컨테이너 기반으로 구축되고 있다. 컨테이너 기술은 기존 리눅스에서 제공하는 Cgroups, 네임스페이스 등을 활용하여 별도로 격리된 OS 환경을 제공하며, 일종의 경량 가상머신의 효과를 누릴 수 있다. 기존 가상 머신 기술과 달리 별도의 하이퍼바이저를 거치지 않고, OS에서 제공하는 기본 기능만으로 격리된 OS 환경을 제공하기 때문에, 실행 속도가 기본 OS와 유사할 뿐만 아니라 사용하기도 간편하다.

도커(Docker)와 같은 컨테이너 기술이 보편화되면서, 수많은 컨테이너에 대한 오케스트레이션 기술도 오픈 소스 프로젝트로 등장했다[8][9]. 또한 최근에는 ARM 환경에 포팅한 사례가 등장하여 라즈베리 파이에도 쿠버네티스 클러스터를 구축할 수 있다[10].

그림 5. 라즈베리 파이로 구현한 플랫폼

ETRI 내부의 오픈 소스 활동의 일환으로 최근 진행한 오픈 소스 기반 테스트베드 구현 사례 중 하나로, 라즈베리 파이를 활용하여 쿠버네티스 클러스터를 구축하고 SDN으로 네트워크를 제어하도록 만든 것이 있다.

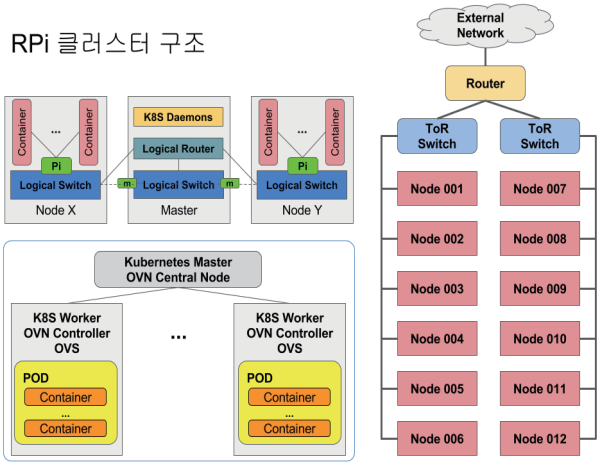


그림 6. 라즈베리파이 기반 쿠버네티스 클러스터 구조

프로토타입으로 12대의 라즈베리 파이 3 모델 B를 두 개의 스위치로 묶었으며, 쿠버네티스 기본 네트워킹 이외에 SDN을 통해 네트워크를 융통성있게 제어하도록 Open vSwitch(OVS)를 설치했다〈그림 x4〉〈그림 x5〉. 라즈베리 파이는 얼마든지 추가할 수 있으며, 스위치와 연결하는 구조는 일반 서버와 비슷하게 처리하면 된다. 라즈베리 파이 기반 쿠버네티스 클러스터에서 활용한 오픈 소스 플랫폼은 크게 도커, 쿠버네티스, OVS로 구성된다. 최근에는 도커와 쿠버네티스가 ARM 버전으로 포팅되어 있어서, 관련 사이트를 참조하면 어렵지 않게 라즈베리 파이에 쿠버네티스 환경을 설치할 수 있다[10][12]. 참고로 기본 라즈비안 OS에 도커를 별도로 설치해도 되지만, 라즈비안에 도커를 기본으로 설치한 OS 이미지를 사용하면 설치 과정을 한 단계 줄일 수 있다[11]. OVS 역시 오래전부터 리눅스에서 기본적으로 지원되는 모듈이므로, 라즈베리 파이에 설치하는 것은 어렵지 않다. 다만, 2016년에 새로 추가된 가상화 모듈인 OVN을 사용하려면 OVS 2.6 이상을 설치해야 하는데, 2016년 11월 기준으로 데비안 계열 리눅스 패키지 시스템에 올라온 OVS는 이보다 낮은 버전이므로, 라즈베리 파이 위에서 직접 소스 빌드를 해야 한다.

이렇게 구축한 쿠버네티스 클러스터는 물론 대규모 데이터센터용 시스템과는 성능면에서 비교할 수 없지만, 소프트웨어 인터페이스 측면에서는 유사하기 때문에 적은 비용으로 많은 수의 컨테이너를 관리하고 네트워킹하기 위한 연구 개발 실험을 간편하게 수행할 수 있다. 비싸고 공간을 많이 차지하는 장비를 모두 갖추지 않고도, 어느 정도 개발 작업을 진행할 수 있는 환경을 제공하는 셈이다.

Ⅳ. 결 론

오픈소스는 우리의 연구 환경을 바꿀 수 있을 것인가? 또 우리의 연구 분야를 확장해 나가는 데 기여할 수 있을 것인가?

우리는 지금까지 기술개발과 표준화 활동이 그다지 긴밀하게 협업이 이루어지지 못하였다. 그것은 '표준화 연계 R&D' 또는 'R&D 연계 표준화'의 필요성이 매번 강조되어 왔음에도 불구하고 우리가 협업 플랫폼을 공유하지 못했기 때문인데, 오픈하드웨어 기반의 플랫폼은 이의 실현가능성을 높여줄 것으로 생각된다.

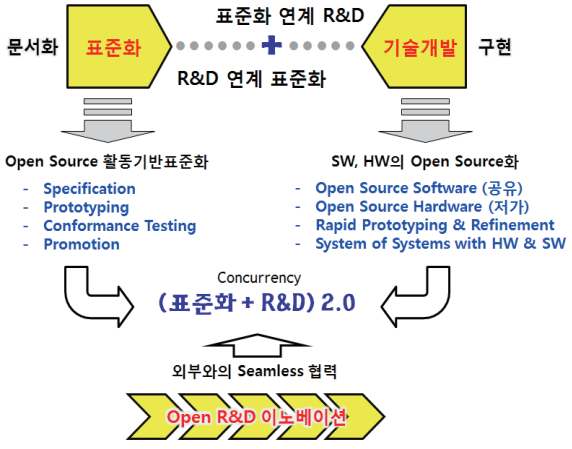


그림 7. 표준화 연계 R&D 패러다임

연구개발의 경쟁력은 아이디어를 얼마나 빠르게 검증하고 완성도 높은 결과를 도출하는 가에 달려있다. 이 과정에서 내외부가 서로 협력할 수 있는 체계가 구축되어 Open R&D 이노베이션이 이루어질 수 있다면 커다란 시너지가 창출될 수 있을 것이다.

R&D 플랫폼을 구성하는 하드웨어 가격이 매우 저렴하므로 원하는 사람은 누구나 이러한 플랫폼을 마련하고, 자체적으로 연구를 수행함과 동시에 필요에 따라 이들을 서로 연결하여 보다 큰 테스트베드로 확장하는 것이 가능할 것이다. 〈그림〉에서 보여주고 있는 라즈베리파이로 구현한 플랫폼은 전체 가격이 100만원을 넘지 않는다.

ETRI 표준연구본부에서는 오픈하드웨어 기반 R&D 플랫폼에 대한 위의 결과를 공유하고, 이를 기반으로 내외부 연구협력을

통해 오픈 R&D 이노베이션을 추진해 나가기 위한 구체적인 방안을 모색하고 있다.

Acknowledgement

본 연구는 "ETRI R&D 표준화 전략 및 기반 구축 연구"를 통해 OpenHaSo AOC (Autonomic Open Community)의 자율적 연구활동으로 수행되었음.

참 고 문 헌

[1] 유재필, "오픈소스 하드웨어 플랫폼 (OPHW)동향 및 전망," Internet & Security Focus 2013 8월호, pp24 – 50
[2] Open Source Hardware (http://www.oshwa.org)
[3] Open Source Hardware Association, "오픈소스 하드웨어의 원칙," (http://www.oshwa.org/definition/korean/)
[4] Wikipedia "Open-source hardware," (https://en.wikipedia.org/wiki/Open-source_hardware
[5] Kickstarter.com "ONetSwitch: Open Source Hardware for SDN," (https://www.kickstarter.com/projects/onetswitch/onetswitch-open-source-hardware-for-networking?ref=nav_search).
[6] 이성원 "오픈소스 소프트웨어 관점 기반 SDI 발전 고찰," 공개SW포탈 2016, (http://www.oss.kr/?mid=oss_repository12&category=642386&listStyle=webzine&page=1&document_srl=642387
[7] 김종원, "융합형 소프트웨어-정의 인프라로의 혁신이란?" 한국통신학회 (http://www.kics.or.kr/Home/UserContents/20161208/161208_093243363.pdf
[8] http://kubernetes.io/
[9] https://www.docker.com/products/docker-swarm
[10] https://github.com/luxas/kubernetes-on-arm
[11] https://github.com/hypriot/
[12] https://www.docker.com

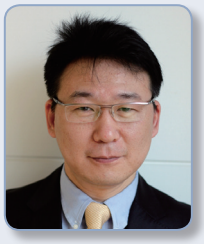
약 력

함 진 호

1982년 한양대학교 전자공학과 학사
1984년 한양대학교 전자통신공학과 석사
1998년 한양대학교 전자통신공학과 박사
1984년~현재 한국전자통신연구원 표준연구본부 책임연구원
2010년~2010년 인터넷미래기술연구부장
2011년~2013년 표준연구센터장
2013년~2013년 청와대 정보방송통신비서관실 선임행정관
2013년~2014년 전략기획본부장
관심분야: 정보통신표준화, ICT R&D 전략, 메이커운동, ICT 기반 교육환경

남 기 혁

2002년 고려대학교 컴퓨터학과 학사
2004년 고려대학교 컴퓨터학과 석사
2004년~2014년 한국전자통신연구원 표준연구센터 선임연구원
2014년~현재 주식회사 프리스티 근무
관심분야: IoT, SDN, Formal Methods



김 형 준

1988년~현재 한국전자통신연구원 (책임연구원)
2012년~현재 사물인터넷포럼 표준분과위원장
2013년~현재 한국전자통신연구원 표준연구본부 본부장
2014년~현재 ICT DIY 포럼 운영위원장
2015년~현재 ITU-T SG13(미래네트워크) 부의장
2015년~현재 ITU-T SG20(사물인터넷 & 스마트시티) 부의장
주요 관심분야: ICT DIY, 미래네트워크, 사물인터넷