How to Derive Abstraction for Software Defined Infrastructure and Software Defined Exchange
A White Paper for Software Defined Infrastructure / Software Defined Exchange Workshop

Kuang-Ching Wang, Clemson University

Researchers have demonstrated engineering feasibility for software defined infrastructure (SDI) and software defined exchange (SDX) through research prototypes over NSF GENI and other future Internet testbeds around the world. The community surely knows how to plumb a diverse range of compute, storage, network, and other instruments into an interoperating system via software and in virtualized slices if the instrument is capable of virtualization. The value proposition of SDI/SDX is clear with many use cases being frequently discussed.

What is not yet clear is, in the real world, how would SDI/SDX be realized. If SDI, as it is envisioned, to be a holistic environment encompassing everything, everywhere that can be controlled by software via a programming interface, then it is fundamental expectation that the elements that make up SDI be of different administrative ownership, have different usage policies, and accepts and serves different users. The same characteristics apply to such elements today – so what's new? In my opinion, the really new challenge is the scale and/or complexity of the potential ways any elements may be programmatically customized to meet respective application needs in this new paradigm. SDX, as a logical exchange point for SDIs, inherit the same potentials and challenges.

So how do we expect applications to program the SDI/SDX? Today, pioneering SDI/SDX researchers either choose to program every aspect of the SDI via low level networking interface (e.g., OpenFlow) and computing interface (e.g., choice of server type, operating system) or choose to leverage simpler sets of service options provided by an underlying operating system based on a supported abstraction. So how do we derive the abstraction? If SDI presents an opportunity for applications to push the envelope of what's possible, it is then reasonable that applications be the ultimate driver of the abstractions that best capture their needs. The looming challenge, in a world where we have all been too used to developing applications for a relatively rigid network given as is by the network operators, is to rethink the abstractions for SDI truly from the applications' perspectives instead of the underlying infrastructure. Given the nearly inexhaustible genres of applications to be had, it is clearly expected that there be a multitude of abstractions and they be as extensible as applications evolve. In addition, as applications are increasingly personalized, the abstractions would clearly address the need for such individualized customizations, suggesting the scale and complexity of programmability expected of the SDI/SDX.

The shift of "control" of the infrastructure to applications also brings challenge to network services such as security and traffic engineering. Mobility of application users adds to the challenge. In the new paradigm, the ability to invoke services such as security via firewalls should not be dictated by where and how the operators set up firewall devices. Instead, applications would inform the SDI/SDX the security needs via an aptly structured abstraction, and the SDI/SDX should fulfill the needs accordingly. The emerging network function virtualization techniques can be one potential solution to address this need. The design of any solutions, however, should be guided with an exploration of the various potential abstractions to describe such personalized, application-driven needs.

Kuang-Ching (KC) Wang as PI for a series of NSF projects (GENI OpenFlow, GENI WiMAX, GENI Cinema, EAGER on "Mobile Gigabit Wireless Access", CC-NIE, and NSFCloud) has led the Clemson team to deploy GENI SDI on campus and conduct experiments stitching compute and network elements over national and international topologies. Wang holds the joint position as Networking CTO for Clemson IT to advise campus strategies in this era of change. Wang has worked with Stanford ON.Lab and Big Switch Networks during his recent sabbatical developing SDN solutions.