If GENI is a Programmable Architecture then

What (Real-Time) Instruction Set Architecture should GENI have?

Azer Bestavros

Computer Science Department
Boston University

best@cs.bu.edu

A Position Statement

NSF Workshop on Real-Time GENI
February 2006

Conceptually, if one considers that users of GENI [1] are in effect providing input to the control plane of GENI, then one can view such a process as *"programming"* the GENI machinery. For example, one can argue that an activity such as specifying QoS requirements or expectations from the underlay is akin to a programming exercise.

The difficulty in programming the envisioned GENI control plane is that what is being programmed is amorphous due its scale as well as to the emergent behaviors that come about due to its open nature. In many ways, the difficulty in defining the interface between the envisioned GENI architecture and its user base is akin to defining an *"Instruction Set Architecture"* (ISA) for GENI. As we know from the evolution of ISA for traditional CPU architectures, the development of an ISA is a balancing act between efficiency and expressive power.

Today, and to a large extent, network control suffers from the same lack of organizing principles as did programming of stand-alone computers some thirty years ago. Primeval programming languages were expressive but unwieldy; software engineering technology improved not only through better understanding of useful abstractions, but also by automating the process of verification of safety properties both at compile time (*e.g.*, type checking) and run times (*e.g.*, memory bound checks). What programming languages have done to software engineering is to force programmers to adopt a disciplined approach to programming that reduces the possibility of "bugs" and by making it possible to mechanically check (whether at compile time or run-time) for unacceptable specifications/behaviors. In many ways, this was done at the expense of reducing the expressive power given to programmers. High-level programming languages do not afford to programmers the same expressive power that assembly language does (*i.e.*, there are programs that one can write in assembly language that one cannot write in Java).

High-level abstractions that restrict expressiveness are not unique to programming languages, they are certainly the norm in Operating Systems, whereby programmers are allowed to interact with (say) system code and resources in prescribed (less expressive) ways. This loss of expressive power is precisely what has enabled us to deal with issues of scale of software artifacts and systems. Yet, the same has not yet materialized for network management and control. Along these lines, the same kinds of benefits in dealing with issues of scale and complexity could find their way into real-time network management and control if we adopt a more "disciplined" approach -- an approach that:

(1) confines the ability of real-time users of GENI to "program" the network, and
(2) allows for compositional analysis and implementation

For example, the notion of expressive power is tightly related to the notion of providing users of GENI interesting in real-time networking with "QoS knobs" – what expressive powers we give to users of GENI (*e.g.*, real-time application developers) is clearly related to what *"knobs"* or parameters of the control plane of GENI we expose, let alone allow users to change.

Finding the *"right"* balance of expressive power and the resulting timeliness guarantees, and mapping such expressive powers to GENI mechanisms is the challenge. What we need is a *"disciplined"* approach for trading off expressive power for real-time characteristics. Such disciplined approaches exist – *e.g.*, using network calculus, statistical scheduling and hierarchical scheduling theories, control theory ... This is exemplified in our work in the iBench initiative at Boston University [2,3].

References

[1] *Global Environment for Network Innovations (GENI).*
http://www.geni.net

[2] *The iBench Initiative.*
http://www.cs.bu.edu/groups/ibench

[3] Azer Bestavros, Adam Bradley, Assaf Kfoury, and Ibrahim Matta. *Typed Abstraction of Complex Network Compositions.* In Proceedings of ICNP'05: The 13th IEEE International Conference on Network Protocols, Boston, MA, November 2005.
http://www.cs.bu.edu/~best/research/papers/icnp05.pdf