The GENI Book

Rick McGeer • Mark Berman • Chip Elliott
Robert Ricci
Editors

The GENI Book

Springer

Editors
Rick McGeer
Chief Scientist, US Ignite
Washington, DC, USA

Mark Berman
GENI Project Office
Raytheon BBN Technologies
Cambridge, MA, USA

Chip Elliott
GENI Project Office
Raytheon BBN Technologies
Cambridge, MA, USA

Robert Ricci
School of Computing
University of Utah
Salt Lake City, UT, USA

ISBN 978-3-319-33767-8 ISBN 978-3-319-33769-2 (eBook)
DOI 10.1007/978-3-319-33769-2

Library of Congress Control Number: 2016948701

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

This book is dedicated to the families of the coeditors: Karen and Sean, Wu and Kashi, Emily, Samantha, and Libby, and Donielle and Michaela, whose unflagging support, perennial grace, and unending patience for workaholic husbands and fathers made both GENI and this book possible. We say thanks often—but we can't say it often enough. So, from each of us to each of you: thanks again. This book is for you.

Introduction

Background: Why GENI?

GENI represents the third wave, following the Grid and the Cloud, of the integration of the network into the computational infrastructure. The first wave, the Grid, focused on the application of distributed computing resources, typically supercomputer sites, towards the solution of a single problem. Essentially, it was an extension of batch processing to multiple sites, to more efficiently use large computing resources. It emerged in the late 1990s and was rapidly extended from scientific to business processing. The Cloud is of course quite familiar, and it refers to two dominant themes. The first is the per-hour rental of virtual machines or other computing resources; the second is the transfer of traditional desktop and enterprise applications to a server accessed over the network, with the Google office suite being perhaps the most prominent example. Of course, new applications are enabled by the Cloud that were unimaginable for the disconnected desktop. Media sharing is a prominent example of this class.

GENI differs from the Cloud and the Grid in that it is a platform for *distributed* applications. A distributed application differs from a Cloud application in that the network is central to the distributed application; it literally cannot exist without the network. While a Cloud application—such as, for example, Google Docs—logically runs on a single computer which happens to be accessed over the network, a GENI application or service can only run in a number of computing environments, geographically dispersed. The most prominent simple examples of this class of application are Content Distribution Networks, Distributed Storage Systems, multicast overlays, and wide-area collaborative exploration and creation systems, and collaborative gaming. The distinctive feature of these systems is that they require geographic distribution for one or a combination of a number of reasons. Perhaps the simplest of these reasons is resilience against local failure. In addition, some applications are inherently distributed, often because of the realities of geographically distant end users and data. Inherently distributed applications often require geographically distributed computing infrastructure to support high bandwidth or low latency to end users and data sources.

The central point about GENI is that the network becomes, not just a way for the user to access an application, but the central component of the application itself. This doesn't require just a different sort of computational platform; to be really effective, the application must have a different kind of network. GENI is the network that undergirds distributed applications, and it is a characteristic of the next generation of computational infrastructure.

Today's network is regarded as a network of simple pipes which carry bits between users and remote applications. The network for distributed applications is far richer and more complex; it consists of a large network of computing elements, and programs move seamlessly between these elements to provide service where required.

Though this sounds exotic, in fact it is simply a different assemblage and deployment of current Commercial-Off-The-Shelf (COTS) hardware and software. To a first approximation, what the developer sees is nothing more exotic than a collection of Linux VMs and containers, interconnected by a more-or-less standard network. However, she is able to allocate VMs and containers in specific places, not simply "somewhere in the Cloud," and she is able to configure the topology and priorities of the network between them. Simply put: she is able to design her own, application-specific, continent- and eventually world-wide network, deploy her application across it, and do so in a matter of moments.

This is an entirely new idea of computational infrastructure, though it is made from standard components. Up until now, the network and the computational service delivered over it were regarded as entirely separate components. The application writer had little control over the network topology, and could only influence packet delivery through the choice of transport protocol and some edge tweaking. Conversely, network engineers regarded the computational devices at the edge as foreign soil. The apotheosis of this attitude was found in the design of Content-Centric Networking. At an application level, CCN was easily achieved as an application-level protocol overlay on Content Distribution Networks. However, the CCN community spent an enormous amount of effort putting content information into the packet header, so that the network equipment could process it. It is a reasonable question on whether the performance penalty for doing content-based routing at the application level was sufficient to warrant the effort to do the application at lower levels of the protocol stack. However, the answer to this question is highly dependent on how tightly interwoven the network and application layers could be. If an application designer can control *where* the application points-of-presence are, and how application packets are routed from the user's host to the nearest application POP, the need to drive the application into the network stack is lessened.

As that example illustrates, there are two brutal realities of the computational infrastructure: network equipment can't be programmed, and computers can't forward packets quickly, and attempts to do either are deeply unnatural. This was ultimately why the ActiveNetworks program of the 1990s failed. This has led the networking community to ever-more complex control protocols to permit intelligent packet handling. But the only reason for intelligent packet handling is the relatively long distances packets must traverse between source and destination; a distributed cloud radically shortens that distance, and thus the demand for network equipment to perform functions better performed by a computer. In sum, the GENI infrastructure with distributed applications leads not only to more effective applications but also to a simpler network.

This overall design of a network, with ubiquitous standard computational components, is seen in many other places. Fifth-generation wireless networks ("5G") is an excellent example. The goal of 5G is gigabit bandwidth and millisecond latency to the wireless device. Of course there is no magic; the physics of wireless devices are well known and the coding schemes are close to the information limit. The only way to achieve the orders of magnitude in performance improvement anticipated in 5G is to radically change the network architecture, and this is exactly what the proposals in gestation at the various nations do. Specifically, all 5G wireless architecture proposals combine very small cells ("picocells") with a computational point-of-presence at the base station. This is the GENI architecture, again; in this case, the distributed applications are serving wireless devices.

The Network Function Virtualization movement in the telco industry is a similar example to the deployment of the GENI architecture. NFV was inspired at least in part by the deployment of carrier Content Distribution Networks, such as CoBlitz. The overarching architectural idea is to replace dedicated hardware with software running in virtual machines. This necessarily means deploying virtual machines over a distributed network infrastructure.

All of these similar architectural initiatives drive from a secular trend; the dramatic and continuing decline in the costs of computation against communication. The chart in Fig. 1, taken from Chap. 20, "The Ignite Distributed Collaborative Visualization System" shows the ratio of the price of a gigaflop of computation vs. a megabit/second of bandwidth. As can be seen from the figure, the ratio has declined from about 10 in 1998 to about 0.1 today, a decline of roughly two orders of magnitude. The most direct explanation for this trend is given in Chap. 20: point infrastructures such as computation follow a technology curve, whereas linear infrastructures follow an adoption curve, and the latter must always trail the former.

Paradoxically, as computation becomes more prolific and widespread, communication becomes much more of a dominant consideration in system design. This is because communication becomes the bottleneck in system performance. This is a secular trend throughout the computing industry, from chip design through, in our case, redesign of the Internet. In the case of chips, this has seen the rise over the past decade of multicore architectures and GPU-based vector computation, as increased parallelism becomes the performance driver rather than increasing clock rates. In single-server and data center systems, it has led to the redesign of the server around high-bandwidth memory systems and the data center around highly parallel massive data set searches and manipulations [1, 2], with an emphasis on Terasort rather than Linpack as a benchmark. This involved a radical change to both the memory hierarchy architecture and the design of very high-bandwidth,

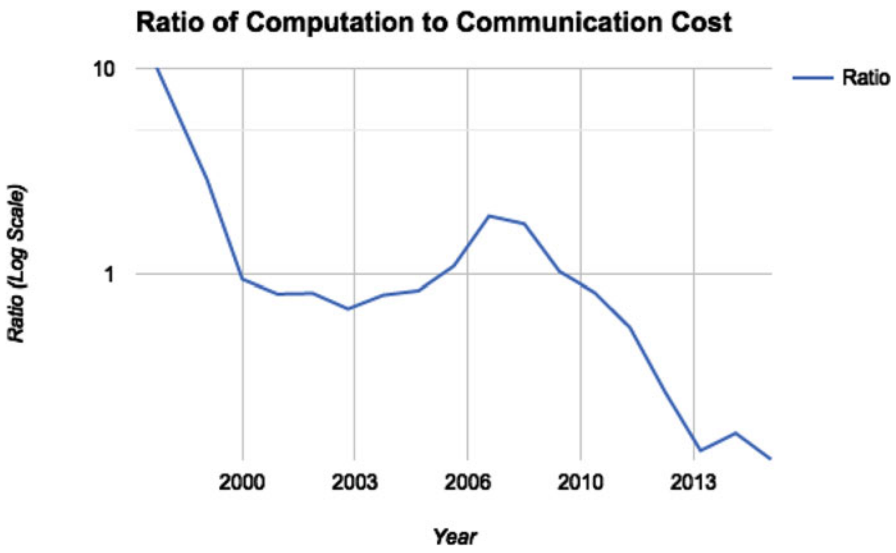

Fig. 1 Ratio of computation to communication cost

low-latency data center networks [3]. In the case of the wide-area network, it is the architecture described in this book: ubiquitous computational points-of-presence with a programmable network between them.

This redesign of the Internet architecture largely leaves the data plane untouched, and in fact radically simplifies the control plane. In fact, the obstacles to its adoption are largely cultural, social, and political rather than technical. We need to rethink our ideas about computation, communication, and data and information storage. Right now, any user of the Internet can tax the communication resources of almost any enterprise or institution; however, access to those institution's computing resources is tightly guarded. There are reasons other than cost, of course, but the dominant reason for this is because our computing systems grew up in an era of time-sharing, where computing was expensive and guarded. Access control was built into the systems from their inception; conversely, communication was unprotected and clumsy access controls retrofitted after the fact. From a cost perspective, this dominant theme of protecting computation but leaving communication open is exactly backwards.

As mentioned, there are other considerations, primarily data security, integrity of the computing environment, and fears of malicious use. But the Cloud has largely overcome those objections: an enormous number of enterprises entrust their data to third-party Cloud storage and do their computing on virtual machines running on the same hardware as an unknown and untrusted third party. A large number of enterprises, universities, and governments outsource their basic IT functions to third-party providers such as Google Apps for Enterprise. It would be an odd CIO indeed who worries about what a student might do with a VM, but will happily offload ERP functions to a cloud provider.

In sum, communication costs a lot more than computing, and we know a lot more about securing computing than we do about securing communication. It's time for GENI's Distributed Cloud.

How Did GENI Come To Be?

As usual, it started with the hackers. Come, be it admitted: academic computer scientists don't do new apps. We do exploit the properties of new technologies to come up with new infrastructures (see, for example, RAID [4] and NOW [5]). But by and large, computer scientists take services and applications hacked up in a hobbyist or commercial setting and build robust, scalable versions of the application or service.

So in the late 1990s people started to exploit the Internet, and a new breed of service known as "peer-to-peer" was born. It was initially popularized by the Napster file-sharing service, but its implications as a communications medium rapidly became apparent. Only a couple of years after Napster was founded, the first wide-area scalable indexing and storage system was devised [6]. A host of implementations followed, along with a large number of distributed applications and services: wide-area robust storage systems, content distribution networks, overlay multicast trees, etc.

This led to an immediate problem: how does one deploy such a system, at scale? In 2001, there was no platform available to deploy these new classes of systems. Rather, what was happening was that researchers were calling up their friends at other institutions, getting accounts on machines at their institutions—with heterogeneous configurations, different software installations, and so on—and then running an experiment. A system that took a few weeks to write might take months to deploy and test.

At an underground meeting at NSDI 2002, a group of researchers led by Larry Peterson of Princeton and David Culler of UC Berkeley devised a new infrastructure to serve as a community testbed. Each institution would agree to devote 2–3 $\times$ 86 servers to a community testbed, which would be centrally managed. To permit each researcher to create his own environment, nascent virtualization technology—Linux VServers—was employed to offer very lightweight virtual machines. David Tennenhouse, then head of Intel research, and Patrick Scaglia, who led the Internet and Computing Platforms division of HP Labs, agreed to form a consortium to fund the platform and grow it to several hundred sites worldwide. And the world's first Distributed Cloud, PlanetLab, was born.

PlanetLab grew rapidly, eventually reaching its current size of 1350 nodes at over 700 sites worldwide. More impressive was its immediate impact on the systems community; the vast majority of SOSP 2003 papers cited PlanetLab experiments just a year after the testbed was first built.

In early 2001, Jay Lepreau of the University of Utah and his staff and students devoted a cluster to network experimentation. The problem the Utah group was addressing was both similar and not to the problem addressed a year later by PlanetLab: the need to do short-run controlled experiments on new network protocols and services. Their Emulab platform became the world’s first Cloud. It differed then and differs now from standard Clouds. Users are able to request hardware as a service, not simply virtual machines, and are able to finely control the emulated network between their nodes. As a result, it immediately became the premier experimental platform for controlled experiments on distributed systems and network protocols, and remains so today. It is described in detail in Chap. 2.

In September 2003, Dipankar (Ray) Raychaudhuri of Rutgers and his staff began the ORBIT program, a large-scale open-access wireless networking testbed for use by the research community working on next-generation protocols, middleware and applications. The ORBIT project continues to this day and has extensions for software-defined radio elements. Like Emulab, it is a shared testbed. Users log in to the ORBIT portal, and then construct an experiment, typically over ORBIT’s 400-node (20×20) indoor radio grid facility. The testbed also includes an outdoor “field trial system” intended to support real-world evaluation for protocols validated on the emulator, and for application development involving mobile end users.

In 2005, UC Berkeley and the University of Southern California Information Sciences Institute collaborated to build a shared state-of-the-art scientific computing facility for security experimentation, the cyber DEfense Technology Experimental Research Laboratory (DeterLab). Based originally on the Emulab software stack, DeterLab has introduced a number of innovations to enhance scalability, reproducibility, and control of user experiments. Of course, since DeterLab is security focussed, some of its principal innovations are to ensure protection and isolation of experiments and protection of the world from running experiments. DeterLab offers security researchers the ability to “observe and interact with real malicious software, operating in realistic network environments at scales found in the real world.” In other words, this is a facility where monsters are observed and experimented on; and so a primary concern, executed with enormous care and great success over more than a decade, is keeping the monsters safely penned while researchers discover how to neutralize them.

Shared experimental facilities such as ORBIT, Emulab, and DETER start from an economic and democratizing rationale—these facilities permit researchers from any institution to conduct experiments on best-in-world facilities, and it is far more efficient and effective for a funding organization to build a large shared facility rather than many small facilities. Not only does this permit researchers to run much larger-scale tests than would otherwise be possible, there are significant economies of scale. There have also been two major scientific benefits. The first is reproducibility. The availability of shared testbeds enables experimenters to report reproducible results which encourage subsequent validation: the test and the experimental facility are accessible by everyone. Moreover, for each of these facilities, simply running the facility and providing new scientific capabilities has been in and of itself a fecund source of research problems.

By 2006, the successes of these platforms were clear to the systems community and the National Science Foundation. Virtually every major experimental and research system built used one or more of these testbeds. In fact, use of at least two was the common case, because the platforms had complementary strengths. Emulab was an ideal system for short-run controlled experiments on new network systems and protocols in a laboratory setting. DETER, though similar to Emulab, had added crucial features to permit safe testing of security protocols, particularly under malware attack. PlanetLab was designed for long-running services and observations of services in the wide area.

However, Emulab, PlanetLab, and DETER had become victims of their own successes. By 2006 all three testbeds were under significant strain due to enormous demand. It wasn't uncommon for researchers to wait days or weeks to get free machines on Emulab or DETER, particularly as major conference deadlines approached. Because PlanetLab offered lightweight virtualization technology, its oversubscription did not appear as waiting times. But enough slices were active on the PlanetLab testbed at any time that load averages on PlanetLab machines could be over 20.

The systems Computer Science community then began to design a successor to these testbeds. The new system had to meet four major goals:

- Incorporate the controllability and flexibility of Emulab and DETER for short-run controlled experiments.
- Incorporate the geographic distribution of PlanetLab for long-running services and applications, particularly end-user-facing applications such as CDNs and multicast overlays.
- Incorporate the wireless aspects of ORBIT.
- Offer fine-grained control of the network and a principled and architectural approach to software control of the L2 and L3 networks.

Over the period 2006–2007 a group of 50 leading academic computer scientists in six working groups designed this system, producing a working prototype design for the National Science Foundation. In 2008 the NSF issued a call for a GENI Project Office (GPO) to manage the development of a prototype of the GENI system, which was won by BBN Technologies. In 2009, BBN led a community effort to develop this prototype, issuing contracts to universities and research organizations in the systems community to develop GENI.

Simultaneously with this was a happy Black Swan event—a revolutionary new technology, Software-Defined Networking. This concept, and its concrete realization, OpenFlow, grew from the Ethane project at Stanford University. Its most basic concept was that a software controller would load the routing tables of a network of L2 switches, permitting fine-grained software control of packet forwarding and QoS. This offered the key last piece that had been missing from the precursors of GENI: integration of the network into the computational infrastructure. OpenFlow immediately became a key component of the emerging GENI.

GENI's Community Development Approach

The entire community recognized GENI as a high-risk endeavor from the outset. At the time the GPO was initially stood up, it was by no means clear that GENI was technically feasible (or even well defined). Accordingly, the GPO chose a spiral-development approach to development, incrementally building, assessing, and redesigning GENI on a continuing basis, with a nominal spiral duration of one year, punctuated by three GENI Engineering Conferences (GECs) annually. Open to the interested public, the GECs provide impetus for community debate, information exchange, and development deadlines.

The GENI community embraced spiral development as a strategy to continuously confront the most pressing questions—technical and programmatic risks—of the day, with successive spirals addressing a sequence of vital questions. The interactions of dozens of development teams and an ongoing design and development effort driven by thrice-annual community meetings set the stage for rapid, if slightly raucous, progress. This community approach also gave rise to one of GENI's central execution strategies: whenever possible, pursue multiple implementations simultaneously.

Time period	Burning question	Key tactics
Spirals 1 and 2 (2008–2010)	"Is GENI technically feasible?"	Control frameworks and slicing
Spirals 2 and 3 (2009–2011)	"Can GENI be built at adequate scale with reasonable cost and effort?"	"GENI-enabling" equipment, federation, and meso-scale prototype
Spirals 3 and 4 (2010–2012)	"Will GENI be useful for research?"	Research-driven design, community outreach, and "GENI-enabling" tools
Spirals 5 and beyond (2012–)	"Will GENI transform the community?"	GENI racks and international federation

The very first spirals aimed to prove the technical feasibility of core GENI concepts. One such concept was a control framework that could manage multiple, heterogeneous suites of infrastructure. The second was an end-to-end "slice" construct that spanned such heterogeneous suites, interconnecting their diverse virtualization technologies. The GPO organized community projects into competing "clusters" (shown in Fig. 2). Projects then integrated within clusters to achieve four prototype GENI systems by the end of spiral 1. By the middle of Spiral 2, three of the major GENI systems (PlanetLab, ProtoGENI/Emulab, and OpenFlow) were capable of interoperation.

As the first technical hurdles were being overcome, the GENI community also confronted the central programmatic puzzle in GENI—how to afford construction and operation of a set of infrastructure that can support "at scale" research experimentation. The GENI meso-scale prototype presented an opportunity to test

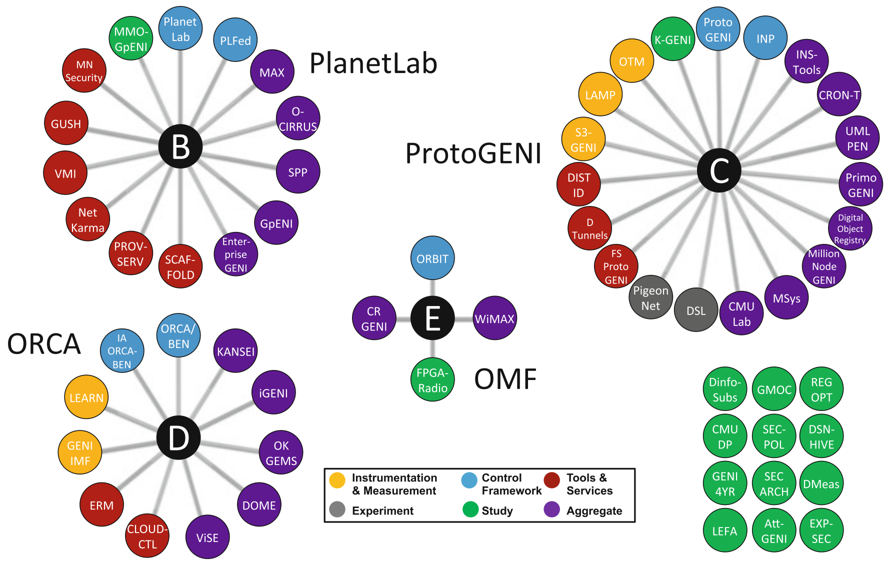

Fig. 2 Early GENI project clusters

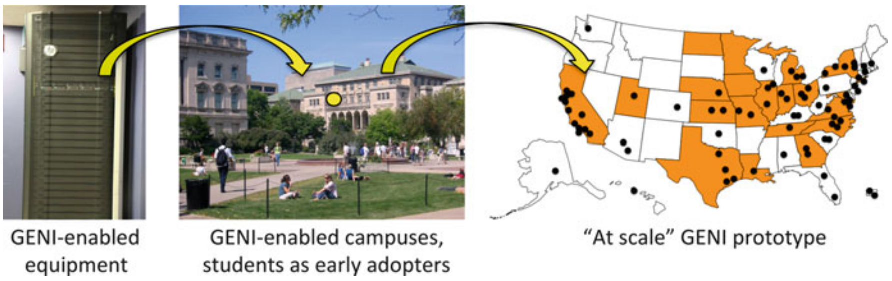
GENI-enabled equipment
GENI-enabled campuses, students as early adopters
"At scale" GENI prototype

Fig. 3 The "GENI-enabled" campus strategy

the strategy of "GENI-enabling" campuses and research networks, as a way to overcome this challenge.

The strategy began by GENI-enabling existing testbeds, campuses, regional and backbone networks, cloud computing services, and commercial equipment. GENI could then incorporate these networks and services by federation, rather than constructing and operating a separate set of infrastructure for experimental research. Figure 3 depicts the plan: first GENI-enable commercial equipment, then use this equipment to create "GENI-enabled" campuses and the national backbones that can run GENI experiments on the same infrastructure as production networks. Finally, federate GENI-enabled campuses and networks to create "at scale" GENI.

The key hardware artifact of spirals 2 and 3 was a "meso-scale" version of this basic approach, spanning 14 campuses and 2 national backbones (Internet2 and NLR). The meso-scale prototype integrated PlanetLab, ProtoGENI, and OpenFlow, with GENI-enabled commercial equipment from HP, Juniper, NEC, and Quanta. While this prototype was functional, it was also finicky, requiring the GPO to work closely with researchers to help them conduct experiments on this early GENI prototype and use their experiences to refine plans for continued GENI development. Importantly, deployment of this prototype generally included involvement from the campus CIO or CTO, establishing a precedent of involving both research faculty and campus IT staff in GENI planning and progress. The meso-scale GENI prototype was eventually decommissioned as the larger, "at scale" GENI deployment subsumed its capabilities.

Experience building and using the "meso-scale" GENI provided a strong indication that an "at scale" implementation would be technically feasible, affordable, and sustainable. The next key question for the GENI community was how to ensure that GENI genuinely opens up major new fields of experimental research.

This question could only be addressed through a feedback cycle where GENI is consistently employed in research experiments and the lessons learned employed in improving future GENI implementations. Beginning with feedback from experiments begun in Spiral 2, joint researcher-developer sessions became a fixture of GECs, and research experiments began to drive GENI's evolving design. Significant outreach and support effort from the GPO, NSF, and the GENI development community encouraged GENI's rapid adoption by researchers in spirals 3 and 4, leading to strong growth in research use. Figure 4 shows a GEC demo night event, where developers and experimenters show off their progress.

Fig. 4 Demo night at GEC16, Salt Lake City, 2013

The “GENI-enabling” approach was also applied to popular research tools. Researchers are accustomed to working with specific tools, and their introduction to GENI was greatly eased by making GENI resources available through these familiar pathways. This approach began with the adoption and interoperation of precursor testbeds like PlanetLab, ProtoGENI/Emulab, and ORBIT and was extended to tools like the OMF control infrastructure and the Open Resource Control Architecture (ORCA) control framework.

As researchers began to experiment with the “meso-scale” GENI, they quickly became aware of its potential, as well as its limitations. Experimenters found great value in the key capabilities of the prototype, including slicing and deep programmability. They wanted a larger-scale deployment, with more programmable computation and network components throughout the GENI network. They needed additional automation to support dramatic growth in the number of simultaneous experiments.

The move to a larger GENI prototype began with basic GENI building blocks. Beginning in 2012, and continuing to the current GENI, campuses are GENI-enabled by deploying GENI racks, optional wireless base stations, and software-defined networks on campus. These resources are connected to a research backbone network and federated into the emerging nationwide GENI, where they are available to the entire GENI research community. The principles involved are consistent with the approach used in the “meso-scale” GENI, but the process is significantly simplified at each campus by the availability of GENI racks. A GENI rack includes computation (cluster of processors), storage, and an OpenFlow switch in a single deployable package, along with its associated control software. The rack provides the campus an entrée into the GENI federation. Additional campus resources, such as a science DMZ, may be federated as well, in keeping with the unique research needs of each campus.

As GENI grew within the USA, similar projects arose around the world. While each of these future Internet and distributed cloud (FIDC) testbeds has unique implementation and management aspects, there is strong motivation both to share ideas and software and to federate infrastructure, and GENI has been a leader in this area for several years. The globalization of FIDC concepts is an unfinished but highly promising chapter of the GENI story.

Organization of the Book

The book takes us through the GENI Project in its lifecycle in five parts. Part I describes the precursors of GENI that led to its development, with detailed histories of ORBIT, DETER, Emulab, and a discussion of the GENI idea from then NSF Assistant Director Peter Freeman. Part II describes the architecture of GENI as a set of control frameworks that interact and present the developer with a picture of a distributed cloud with a programmable network in between cloud nodes and describes how the specific precursors of GENI—PlanetLab, Emulab, and

ORBIT—were adapted into new complementary control frameworks within GENI. These chapters also discuss how new technologies, specifically emerging Cloud technologies and the new capabilities of software-defined networking, were adopted and integrated into the GENI framework and the specific control frameworks which made it up. Part III discusses the deployment of GENI as a nationwide infrastructure. Once the control frameworks were in place, GENI had to be made concrete and real. The control frameworks were integrated and deployed at 50 sites across the United States, in small, extensible clusters: "GENI Racks." These were interconnected by a programmable nationwide layer-2 network, the "Mesoscale Deployment." Once this was in place, GENI was ready to host applications and services. Part IV describes the applications of GENI to our society and profession, and the tools developed to use this infrastructure. GENI is not alone; it is one of several similar efforts worldwide. Part V discusses parallel and complementary efforts in Canada, Europe, and Asia, and the prospects for an international federation.

The story of GENI is far from done. We are now roughly where the NSFNet was in the late 1980s, with a few tens of sites connected by a nationwide backbone. As GENI transitions to the next phase of its life, which we believe will be an era of explosive growth, we recall the words of Vint Cerf as the ARPANET transitioned to become part of the Internet:

> It was the first, and being first, was best,
> but now we lay it down to ever rest.
> Now pause with me a moment, shed some tears.
> For auld lang syne, for love, for years and years
> of faithful service, duty done, I weep.
> Lay down thy packet, now, O friend, and sleep.
> -Vinton Cerf

Washington, DC — Rick McGeer
Cambridge, MA — Mark Berman
Cambridge, MA — Chip Elliott
Salt Lake City, UT — Robert Ricci

References

1. Ranganathan, P.: From microprocessors to nanostores: Rethinking data-centric systems. IEEE Comput. **44**(1) (2011)
2. Ousterhout, J. et al.: The case for RAMClouds: scalable high-performance storage entirely in DRAM. SIGOPS Operat. Syst. Rev., **43**, 4, 92–105 (2009)
3. Al-Fares, M., et al.: A scalable, commodity data center network architecture. Proc SIGCOMM. (2008)

4. Patterson, D. et al.: A case for redundant arrays of inexpensive disks (RAID). ACM Sigmod Record. (1988)
5. Anderson, T., et al.: A case for NOW (networks of workstations). IEEE Micro. (1995)
6. Ratnasamy, S., et al.: A scalable content-addressable network. Proc. SIGCOMM. (2001)

Acknowledgements

In his ballad "Calypso" about the research vessel of the undersea explorer Jacques Cousteau, John Denver penned these lyrics: "Aye Calypso, I sing to your spirit/The men who have served you so long and so well." The spirit of GENI is our colleagues: the hundreds of women and men who worked tirelessly over a decade to take this from an idea to the genesis of the next Internet. Before and above all else, we thank each and every one of them: it has been our great privilege and honor to know and work with each of them. This book is their ballad. Our research sponsors at the National Science Foundation have shown courage and vision and unflagging support over the years. We thank Suzi Iacono, Farnham Jahanian, Peter Freeman, Guru Parulkar, Gracie Narcho, Erwin Gianchandani, Bryan Lyles, Jim Kurose, and Jack Brassil for their leadership at NSF. We have been encouraged and supported by the White House Office of Science and Technology Policy, and thank particularly Tom Kalil. GENI would not exist without Larry Peterson, whose PlanetLab provided much of the technical inspiration for GENI and whose leadership was vital in starting the GENI project at NSF. We thank Larry and his early colleagues, particularly Tom Anderson, Scott Shenker, and Jon Turner.

We need to thank a friend no longer with us. The other large inspiration for GENI was Emulab, and that was the brainchild and passion of our close friend, and Rob Ricci's mentor, Jay Lepreau. Jay, sadly, passed away before this project was properly begun, but his spirit lives on in it, and we hope in us. Thanks, Jay, and we all still miss you terribly.

As we write these thanks, it occurs to us that many names which should be here are not—the list of those we should thank by name is so long that it would take many pages to enumerate them all, and we are certain to forget some. GENI is an enormous effort, and many people in many different roles played key parts in making it happen, and all were cheerful, graceful, and went far beyond the call of duty. This is inadequate, but the best we can do here: thanks. You know who you are, and so do we, and we will be forever grateful. Drinks are on us, the next time we meet.

GENI is supported under cooperative agreements from the US National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this book are the authors' and do not necessarily reflect the views of the National Science Foundation.

Contents

Contributors

Ilya Baldin Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Hadi Bannazadeh Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada

Nicholas Bastin Barnstormer Softworks Ltd. and University of Houston, Houston, TX, USA

Andy Bavier Princeton University and PlanetWorks, LLC, Princeton, NJ, USA

Terry Benzel USC Information Sciences Institute, Marina Del Rey, CA, USA

Jim Blythe USC Information Sciences Institute, Marina Del Rey, CA, USA

Marshall Brinn GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

Charles Carpenter Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Claris Castillo Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Jeff Chase Duke University, Durham, NC, USA

Jim Chen International Center for Advanced Internet Research, Northwestern University, Chicago, IL, USA

Heidi Picher Dempsey GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

Sarah Edwards GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

Ted Faber USC Information Sciences Institute, Marina Del Rey, Los Angeles, CA, USA

Zongming Fei Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Stefan Fischer Institute of Telematics, University of Lübeck, Lübeck, Germany

Peter A. Freeman Georgia Institute of Technology, Atlanta, GA, USA

Fraida Fund NYU School of Engineering, New York City, NY, USA

Jingguo Ge China Science and Technology Network, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

Abhimanyu Gosain GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

James Griffioen Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Paola Grosso University of Amsterdam, Amsterdam, The Netherlands

Chris Heermann Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Matt Hemmings Computer Sciences Department, University of Victoria, Victoria, BC, Canada

Alefiya Hussain USC Information Sciences Institute, Marina Del Rey, CA, USA

Guillaume Jourjon NICTA, Australian Technology Park, Eveleigh, NSW, Australia

Thanasis Korakis NYU School of Engineering, New York City, NY, USA

Robert Krahn Y Combinator Research, San Francisco, CA, USA

Cees de Laat University of Amsterdam, Amsterdam, The Netherlands

David Lary Department of Physics, University of Texas at Dallas, Dallas, TX, USA

Alberto Leon-Garcia Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada

Tong Li China Science and Technology Network, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

Te-Lung Liu National Center for High-Performance Computing, National Applied Laboratories, Hsinchu City, Taiwan

Mon-Yen Luo National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan

Joe Mambretti International Center for Advanced Internet Research, Northwestern University, Chicago, IL, USA

Anirban Mandal Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Rick McGeer Chief Scientist, US Ignite, Washington, DC, USA

Olivier Mehani NICTA, Australian Technology Park, Eveleigh, NSW, Australia

Jonathan Mills NASA Center for Climate Simulation, Goddard Space Flight Center, Greenbelt, MD, USA

Jelena Mirkovic USC Information Sciences Institute, Marina Del Rey, CA, USA

Paul Müller Integrated Communication Systems Lab., Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany

Akihiro Nakao The University of Tokyo, Tokyo, Japan

Hussamuddin Nasir Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Victor Orlikowski Duke University, Durham, NC, USA

Max Ott NICTA, Sydney, Australia

Sergio Rivera P. Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Ronald van der Pol SURFnet, Utrecht, The Netherlands

Thierry Rakotoarivelo NICTA, Australian Technology Park, Eveleigh, NSW, Australia

Dipankar Raychaudhuri WINLAB, Department of ECE, Rutgers University, 674 Rt. 1 South, North Brunswick, NJ 08902, USA

Jeremy Reed Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Martin Reed University of Essex, Colchester, UK

Glenn Ricart US Ignite, Washington, DC, USA

Robert Ricci Flux Research Group, University of Utah, Salt Lake City, UT, USA

Niky Riga GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

Marko Röder Y Combinator Research, San Francisco, CA, USA

Paul Ruth Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Stephen Schwab USC Information Sciences Institute, Marina Del Rey, Arlington, VA, USA

Ivan Seskar WINLAB, Department of ECE, Rutgers University, 674 Rt. 1 South, North Brunswick, NJ, USA

Michael Stanton Brazilian Research and Education Network—RNP, Rio de Janeiro, RJ, Brazil

Vicraj Thomas GENI Project Office, Raytheon BBN Technologies, Cambridge, MA, USA

John Wroclawski USC Information Sciences Institute, Marina Del Rey, CA, USA

Xiongqi Wu Laboratory for Advanced Networking, University of Kentucky, Lexington, KY, USA

Yufeng Xin Renaissance Computing Institute (RENCI)/UNC Chapel Hill, Chapel Hill, NC, USA

Kazuhisa Yamada NTT Network Innovation Lab, Musashino-shi, Japan

Chu-Sing Yang National Cheng-Kung University, Tainan City, Taiwan

Fei Yeh International Center for Advanced Internet Research, Northwestern University, Chicago, IL, USA

Junling You China Science and Technology Network, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

Michael Zink Department of Electrical and Computer Engineering, University of Massachusetts in Amherst, Amherst, MA, USA