Computing as if Infrastructure Mattered

Jean-François Blanchette
Department of Information Studies, UCLA

September 27, 2012

Since its original formulation by Wing in 2006, "computational thinking" [9] has generated considerable interest as an approach that promises to solve a persistent gap: the dramatic expansion of the role of computing in every dimension of society and the general lack of appreciation by the public of its underlying principles. Wing argues this gap must not only be bridged, it must be reversed: the extraordinary success of computing technologies is proof that, from recursion to time-space trade-offs, the conceptual tools developed by computer scientists have broad application beyond mere computing. Indeed, Wing contends the ability to understand and apply fundamental computational principles to a wide range of human endeavors will increasingly count as a core literacy skill for the information age. Computational thinking will thus provide "a basis for lifelong learning of increasingly new and advanced computational concepts and technologies" [5].

While the spread of computational thinking will clearly do wonders to disseminate the *intellectual* achievements of the field, I want to argue that it places insufficient emphasis on the equally significant *social* and *material* contribution of the field, the computing infrastructure. It is this enormously complex system of material resources, institutional practices, economic structures, standards and regulations that provides the existential ground for the provision of computing services. Consequently, it is understanding of the dynamics of that infrastructure that is the essential skill required for a world where, from automobile repair to the humanities, the evolution of countless professional fields has become intimately tied to the evolution of computing itself. Indeed, in pursuing their professional opportunities, students and practitioners from a wide range of disciplines will need to answer a broad range of infrastructure-related questions. For example,

- What data formats and storage strategy offer the best guarantees for long-term preservation of digital assets, given my organization's workflow, existing computing assets, budgetary constraints, and regulatory requirements?

- How can the economies of scale afforded by cloud computing best be leveraged for a given project? What cost, efficiency, and risk trade-offs should be factored in?
- Among the dozens of metadata standards available, which one offers the best return on investment for discovery and access, for a given type of digital resource?
- What new and profitable information services can be built using the Google Books API? How might these services take advantage of advances in interface and sensing technologies — touch, speech, eye-tracking?
- What are the consequences of adopting open-source software for an institution? How does one discriminate between the different types of governance and economic structures that undergird different open-source projects?

In answering such questions, application-specific skills and fundamental principles of computation provide little guidance. Nor do they clarify the technology-related headlines of major newspapers and magazines: the pros and cons of net neutrality, the intensity of codec warfare, or the defining infrastructural work of our time, the shift to cloud computing. To clarify such issues, current and future professionals must have access to the material and technical fundamentals of the computing infrastructure and the principles of its social organization. Indeed, for the graduate professional degrees that are increasingly becoming the educational baseline for the workforce [7], what is required is a set of skills to analyze the complex forces that direct infrastructural evolution. It is such skills that provide the means to anticipate the curve ahead in a continuously moving technological world.

How infrastructure moves

By definition, all infrastructures strive for invisibility. Users are understandably interested in the applications that perform useful services in their lives, rather than in the layers of system abstractions or the physical components that make these services possible. Yet whether it takes place at the level of operating systems, data centers, communication protocols, file formats, standardization, or policy work, the design, planning, and operation of the computing infrastructure is a major professional activity of computer scientists. The remarkable waves of social and technical innovation unleashed by the deployment of the Internet have focused attention on some dimensions of this activity, in particular, the TCP/IP protocols and the governance mechanisms that have guided their design and implementation (e.g., [8]). The enshrinement of the Internet's modular structure as the technical foundation of net neutrality has further underlined the links between infrastructure, public policy, and economic development. These links remind us that infrastructural design is never merely dependent on technical imperatives, but also on sustaining a delicate balance between serving the general interest and maintaining economic competitiveness

among infrastructural stakeholders.

This balancing act is inscribed deep within the very DNA of the computing infrastructure, as it attempts to fulfill two distinct objectives: on the one hand, mediate applications' access to the physical resources of computation—processing, storage, and connectivity—through layered abstractions; on the other hand, multiplex these limited resources so as to efficiently respond to competing demands from applications. The inherent difficulties in simultaneously meeting these two goals set in motion a series of *infrastructural dynamics* — persistence, efficiency trade-offs, scarcity, drift — that gives the computing infrastructure its distinctive evolutionary path.

Infrastructure persists — In spite of the furious pace of IT innovation, the computing infrastructure evolves very slowly. Indeed, the system abstractions that dominate the processing, media, and transport stacks have been around for decades—more than 65 years for the von Neumann model, 45 for the file and the packet. Abstractions persist as they become embodied in hardware (e.g., routers), software (e.g., protocols), and institutions (e.g., programming textbooks). They further persist through the inherent rigidity of modular decomposition, since changes to a module's interface requires renegotiating boundaries with all communicating modules. Long-term persistence provides the stability necessary for economies of scale to take hold. It is such economies that have afforded the von Neumann machine a better cost/processing power ratio than the numerous alternative architectures that for decades have failed to supplant it. At the same time, the high costs of infrastructural investments ensure that computing resources are repurposed rather than merely replaced. The resource stacks must thus compose with different types of materials—serial and parallel architectures, twisted pair and fiber, tape and flash storage—and ensure their backward compatibility. Infrastructural change thus proceeds conservatively through mutation and hybridization, rather than outright break with the past.

Trades-offs all the way down — Modularity is a powerful design strategy: by decoupling abstraction from implementation, it breaks down complex systems in independent yet coordinated organizational units and provides for flexibility in coping with technical change. In doing so, modularity constitutes the primary social and technical order of the computing infrastructure. It is thus remarkable that the costs of modularity are rarely noted upon in the literature. The flexibility modularity brings to abstraction and implementation is always incurred at the price of efficiency trade-offs—e.g., the von Neumann architecture and the serial model of programming it inherently favors. Because of these infrastructural biases, the *pax romana* of modularity is always under threat, under pressure to extract more computational work from the current organization of the stacks. This tension becomes particularly apparent as new types of computational resources require integration within the infrastructure: the shift from singlecore to multicore, magnetic to flash

media, wireline to wireless will reverberate throughout the stacks, as efficiency trade-offs are renegotiated [1]. Abstraction and implementation are thus always in tension, a tension that provides a key entry point for analyzing infrastructural change [2].

Infrastructure manages scarce resources — Data centers already consume 3% of the world's total electrical output, a number that uncomfortably connects computing cycles to coal extraction [4]. The computing infrastructure is however not merely concerned with managing the scarcity of electrical power, but also that of processing, storage, and connectivity. Abstractions not only relieve programmers from the burdens of keeping track of the finiteness of resources, but also from how these are shared amongst competing applications. But sharing a resource also inevitably entails various efficiency trade-offs, favoring some types of applications over others. Reliance on networked computing for an ever broader range of essential services — from telesurgery to intelligent transportation and national security — will require providers to devise policies for prioritizing competing demands on shared computational resources, with much more significant implications than mere jittery music videos.

Slow drift — The computing infrastructure is a constantly evolving system, continuously responding to and integrating growth in size and traffic, technical evolution and decay, new applications, services and implementations, emergent behaviors, etc. This evolution is constrained by the dynamics outlined above — persistence, efficiency trade-offs, and the necessity to share scarce resources. These constraints induce a limited set of possible evolutionary paths, i.e., co-design of layers, encapsulation, insertion of new layers (so-called middleware). Thus, infrastructural development never proceeds from some clean slate, but rather, from the push and pull of competing stakeholders working to shift its evolution in the most advantageous direction. Even the OSI model, the best example of top-down, a-priori modular decomposition of a resource stack, was immediately challenged in the marketplace by the more widely implemented TCP/IP stack. Always only partially responsive to rational control, infrastructural evolution is characterized by drift, opportunity, and improvisation [3].

An infrastructure-centered perspective thus requires students to engage with computing in a manner that fully integrates network economics, standardization, human-computer interaction, modularity, the material resources of computation, regulation, and systems design. In such a perspective, the fundamental principles at the heart of computational thinking are inseparable from the messy work required for their material realization as infrastructure. There are few pedagogical resources to help promote such a view (see [6] for an exception). By and large, computer science textbooks remain primarily interested in abstraction as the fundamental practice of the field. Yet, as a breathtaking proportion of social relations becomes mediated through information technologies, it becomes neces-

sary to think of computing as equally concerned with infrastructure building, with all the real-world complexity and engagement with the social world this entails.

References

[1] Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, and Katherine Yelick. A view of the parallel computing landscape. *Commun ACM*, 52(10):56–67, 2009.

[2] Jean-François Blanchette. A material history of bits. *Journal of the American Society for Information Science and Technology*, 62(6):1042–1057, 2011.

[3] Claudio Ciborra. *From control to drift: the dynamics of corporate information infastructures*. Oxford University Press, USA, 2000.

[4] Gary Cook and Jodie Van Horn. How dirty is your data? Technical report, Greenpeace International, Amsterdam, 2011.

[5] Herbert S. Lin, editor. *Report of a Workshop on the Scope and Nature of Computational Thinking*. National Academies Press, Washington, D.C., 2010.

[6] David G. Messerschmitt. *Networked applications: a guide to the new computing infrastructure*. Morgan Kaufmann Publishers, 1999.

[7] Laura Pappano. The master's as the new bachelor's. *The New York Times*, page ED16, July 22 2011.

[8] Barbara Van Schewick. *Internet Architecture and Innovation*. The MIT Press, 2010.

[9] Jeanette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.