I’ve often mused about the properties of structures that make them scalable. Regularity is often an element, as in geodesic domes. Abstraction is another contributor. An example of this is the layered, hierarchical, and federated structure of the Internet. Sequences of bits (physical transmission) are organized into frames (link structure) that are organized into packets with header and payload structure. TCP organizes sequences of packets into “connections” that deliver abstract sequences of payload bytes (“octets”) and are organized into higher-level abstractions. At each “level” of abstraction, the details become less visible, and emergent structure becomes apparent. Groups of computers (for example, routers) form Autonomous Systems (networks) that share a common property (an “internal gateway protocol” such as IS-IS and an “external gateway protocol” such as BGP4). Encapsulation hides the lower-layer structure in the payload of a “higher”-layer structure.

All of these architectural concepts have the effect of loosely coupling the higher-layer abstractions to each other by hiding details or confining the lower layers’ detailed properties. This loose coupling allows for flexibility and adaptation while avoiding the rigidity and brittleness that tightly coupled systems sometimes exhibit. It also lets the system accommodate various forms of distribution or centralization within the architecture. Standardizing interfaces and protocols allows distinct implementations to coexist and interwork by hiding the details of their differences in the regularity of the form that the information they exchange takes. Standards confer scalability by establishing interoperability without requiring an enormous number of bilateral or multilateral agreements beforehand. A standard is, in effect, like a multilateral treaty among cooperating parties. Federation confers a degree of autonomy on cooperating actors without assigning to one or another a “master” or “slave” relationship.

There is a price to pay for hiding information — namely, optimization. Some performance optimizations would be achievable only if higher layers in an architecture knew more about the conditions lower down. This tends to expose the architecture to some degree of brittleness because it then depends on this information, making it somewhat less general and adaptable. On the other hand, systems that establish a strong self-measurement regime at multiple layers might be able to detect and diagnose anomalous behavior, deliberate attacks, or signs of pending failure. This kind of measurement can involve multiple layers as many operations and management (OAM) designs deliberately gather state information from multiple layers and deliver it to an operational management function. Such tactics range from monitoring bit flow on wires to application-level behaviors and everything in between.

To the extent that design specifics aren’t treated as proprietary trade secrets, we might learn much by studying, measuring, and analyzing the designs of large-scale software (and networking) systems. Finding principles to extract from a number of large-scale systems is also an exercise in abstraction. Design principles emerge through representing and analyzing systems at the right abstraction level: too detailed, and principles are obscured; too high-level, and the same hazard awaits.

Systems architecture and systems engineering deal with these concepts, and it’s my belief that they’re under-emphasized in today’s curriculum. In addition to their application in
Abstraction, Federation, and Scalability

engineering, it seems clear that the roots of design principles must also lie at the heart of what’s called “computer science,” in which we try to understand at a fundamental level how and why complex, artificial systems behave in certain ways. We try to derive rules of thumb or more precise analytic models of ways to predict system behavior at various scales of operation. Much work remains before we can understand how best to design these systems — anyone looking for a dissertation topic will not be disappointed at the richness of this research area.

Vinton G. Cerf is vice president and chief Internet evangelist at Google. Contact him at vint@google.com.

IEEE Computer Society’s Conference Publishing Services (CPS) is now offering conference program mobile apps! Let your attendees have their conference schedule, conference information, and paper listings in the palm of their hands.

The conference program mobile app works for Android devices, iPhone, iPad, and the Kindle Fire.

For more information please contact cps@computer.org