DOI:10.1145/2043174.2043176

To Boost Presentation Quality, Ask Questions

MOSHE Y. VARDI'S Editor's Letter "Are You Talking to Me?" (Sept. 2011) said conference attendees are sometimes unable to follow speakers' presentations and eventually give up trying. So how about if ACM and IEEE would run an experimental conference where session chairs are expected to ask questions during presentations when they themselves lose track or when audience members clearly stop paying attention. Note such an experiment would have to be done without undue disruption and not allowed to reflect on a particular speaker.

The biggest trade-offs would be the extra time presentations might require and the possibility of upsetting overly sensitive speakers. However, they could be addressed experimentally, initially at small, highly technical conferences with flexible break periods and by selecting only expert, personable chairs to manage the sessions.

Robin Williams, San Jose, CA

Author's Response:

I agree that ACM and IEEE conferences should experiment to improve the quality of their talks. Some ideas can be implemented fairly easily, as in, say, asking conference attendees to give anonymous feedback to speakers. However, one must keep in mind that conferences are grassroots operations, and experiments cannot be dictated by association governing bodies. Rather, the effort to improve conference talks must be undertaken by conferences on their own initiative.

Moshe Y. Vardi, Editor-in-Chief

Adopt the End-to-End Principle in Home Networks

To address the user-experience concerns raised in "Advancing the State of Home Networking" by W. Keith Edwards et al. (June 2011), we must first understand why home networks have been so successful despite the very real difficulties cited in the article. In

attempting to do better for users, we might, in fact, do just the opposite. The authors recognized that developers treat networks as opaque infrastructure, which is the fundamental architectural principle that has made the Internet so generative.

Classic telecommunications is the business of providing services like the public switched telephone network, or PSTN. The Internet is a different concept, providing a common infrastructure for all services. Yet the very power of the Internet, which allows us to tunnel through legacy telecom, has also led us to accept the idea that it is just another service, like PSTN.

In the 1990s this was the plan for home computers, too. Working at Microsoft (Jan. 1995), I realized that home networking could be do-it-yourself rather than a service with a monthly bill and restrictions on what we do. I took the approach of removing complexity rather than adding solutions. Windows 98se supported the necessary protocols to "just work." This involved the requirement that the user would not have to buy any service beyond a single IP but share a single IP address. I wanted to use IPv6 so each device would have a first-class presence. But because IPv6 was not available at the time, I used Network Address Translation to share a single IPv4 address.

Rather than make the home network smarter and more cognizant of the particulars of the home, we must honor the end-to-end principle and treat the Internet as infrastructure. Developers would thus be relieved of the impossible burden of having to understand the home environment and its inhabitants. Any number of approaches could coexist.

Today's Internet protocols date from when big computers were immobile and relationships could be defined through fixed IP addresses. To preserve this simplicity, we need stable relationships for our untethered devices. This way, we could address sources of complexity rather than their symptoms.

Bob Frankston, Newton, MA

Fewer Lines of Code for More Results

In Poul-Henning Kamp's article "The Most Expensive One-Byte Mistake" (Sept. 2011), did Ken, Dennis, and Brian indeed choose wrong with NUL-terminated text strings? I say they chose correctly, then and now. The reason C is dying and nobody has used PL/I, Algol, or Pascal for real work for the past 30 years is that C makes it possible to accomplish a lot in a few lines of intuitive code despite requiring little memory or CPU power. Searching and comparing NUL-terminated strings can be accomplished with such short code segments; programmers hardly need a standard library, and code compiles into a few PDP-11 machine instructions. Failing to check untrusted data is fatal in any language.

C allows fast simple code written by competent programmers, and simple code tends to be less buggy and more readable than complex code. For programmers who still want to use address + length strings, such use can be accomplished in just a few lines. There is, of course, the `strlen()` function to measure the string's length and the `fgets()` function to limit how many characters to read into a string from a file.

Sure, copying large strings can run faster with newer hardware if the string lengths are known. This is a trade-off, and programmers can, if desired, use address + length strings in C and even word-align them. For others, there is always "C with Training Wheels," a.k.a. Pascal or Java, if one is in no special hurry for results.

Good programmers write secure code; bad programmers write insecure, buggy code. Good practices are more valuable than "magic" language features. The largest Java application I know is also the buggiest application I know.

Bob Toxen, Atlanta, GA

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

© 2011 ACM 0001-0782/11/12 \$10.00