

Improvements in distributed processing are limited by the pace of change of the design tools used to achieve them.

Vision Becomes Reality

For technologists, highly distributed processing has been as tantalizing a goal as any. Several years ago, a networking company CTO predicted that soon over 128 IP-addressable locations on each person's body would empower humankind to plug into the Internet at "dot.com" speeds. Unfortunately, the financial backing for such technical advancements collapsed with the high-tech boom long before the IPv6 protocol needed for multiple addressable body points could be realized.

That's not to say that we don't have a broad spectrum of distributed connectivity offerings today, ranging from wearable computing and Ethernet-enabled consumer devices to rack-mounted highly available systems and hardware-accelerated deep-packet messaging. JavaBeans, gateways, processor arrays, busses, backplanes, wireless and Infiniband solutions are among the topologies for ubiquitous computing over distributed systems currently in use.

In fact, the topology is the easy part in distributed systems design. The real design challenge lies in moving massive amounts of data through these systems, from process to process to real-time thread and back again.

The modeling environments needed to design complex distributed systems are on the verge of a new standard—UML 2.0. When this standard is finalized by the OMG committee, perhaps within this calendar year, engineers will have a functional design tool for real-time knowledge management that can be built directly into the design and one specifically geared for handling distributed systems.

Currently, there are many operating systems that support applications and infrastructures for large distributed systems—various flavors of UNIX or Windows that act as software applications running on hardware platforms with knowledge of real-time deterministic behavior. Meanwhile, there may be other computing devices within a distributed system which use real-time thread-based operating systems to provide scheduling support for priority computing and interruptions.

As the data move within this mixed, heterogeneous distributed system from node to node, it begs the question: Can data packets moving from one computing environment to the next within a system achieve the requisite transit speeds?

By
Ron Fredericks

Suppose data packets start as digital video camcorder images moving at 30 frames-per-second onto a Windows platform via an ISDN synchronous-protocol modem and then onto a UNIX system serving as a process-based application gateway. The UNIX system, in turn, might send the video data to an automobile's "infotainment" system, which then runs a real-time threads-based display of the original video. In this example, every other system has real-time applications moving the data while the alternately placed systems have only process-model-based applications handling the packets.

Zip, zero, nada

How good are the chances that the final video plays smoothly and accurately end-to-end? Zip, zero, nada if the system has been developed using the limited crop of design tools available today.

Hope is on the way, however, with the advent of the new UML-based tools that will allow designers to incorporate real-time knowledge into system behavior. The impact of real-time response will then be measured during the design, thus increasing the demand for more real-time operating systems capable of handling data end-to-end. With smaller real-time operating systems being built into larger distributed systems, visionaries will again be free to make their wild, futuristic predictions of total-body connectivity - although why we would need IP addressable points all over our bodies has yet to be determined.

Ron Fredericks is a partner manager for Wind River based in Alameda, CA. He has fifteen years of real-time and embedded-systems experience.

