Why are there broadcast-based mechanisms in Windows?

devblogs.microsoft.com/oldnewthing/20050627-00

June 27, 2005

Raymond Chen

Many Windows information mechanisms are based on message broadcasts, among them DDE, WM_FONTCHANGE, and changes in system settings. Why do these mechanisms use broadcasts, when we know that broadcasts can result in the system grinding to a halt due to windows that have stopped processing messages? Because in 16-bit Windows, you didn't have this problem. Recall that 16-bit Windows was co-operatively multi-tasking. When a program received control of the CPU, it could do anything it wanted, knowing that no other programs could run until it explicitly yielded control by calling a function such as `GetMessage` or `PeekMessage`. The downside of this, of course, was that a single hung program caused the entire system to hang, because it wasn't releasing the CPU. The upside, however, was that if your program was running, then you knew, *a priori*, that there were no hung programs in the system. How do you know that? Because if there were a hung program, **it would be running and not you**. If there's only one thing, and you have it, then you know that nobody else is hogging it. Therefore, broadcasting messages was completely safe in 16-bit Windows. You didn't have to worry about non-responsive programs because you had proof that there weren't any. Of course, when the switch to pre-emptive multi-tasking occurred, this assumption no longer applied, but by then it was too late. The broadcast-based model was already in use, and consequently had to be preserved for compatibility reasons. (It would be bad if, for example, Lotus 1-2-3 stopped working on Windows NT because DDE broadcasts were no longer supported. If the Windows NT team had tried that gambit, nobody would have upgraded and Windows NT wouldn't have survived to make a second version.)

On the other hand, given the risks involved in DDE broadcasts, you probably would be better off designing **your** program to not use dynamic data exchange as a data communication mechanism, thereby avoiding the pitfall of message broadcasts. No point contributing to the problem.

Raymond Chen

Follow