The problem with push: can small developers afford it?

By Erica Sadun | Last updated April 12, 2009 8:05 PM

Push was the big buzz this past March. That's when Apple finally announced its long awaited push notification support for the iPhone. Push notifications allow third parties to send instant updates to users, even when applications aren't running. CPU-intensive and possibly battery-consuming operations are run on Web-based servers rather than on the unit itself. For all that consumers might win from this approach, moving those operations off phones may create a huge financial burden for small developers.

Push-based services step in for any kind of program that would normally check for information updates. For example, they might poll for new direct messages on Twitter or respond to sensors for in-home security systems. When new information is available for a client, the server pushes that update through a remote notification system. The notification transmits directly to a device, in Apple's case to a phone, which has registered to receive those updates. When a message arrives, that device can display a text alert, play a sound, or offer to launch a program in response.

According to Apple, battery life is the single biggest reason for endorsing push notification. It lets Apple's iPhone developer agreement disallow background processes. Background processes, in this case, refers to letting more than one third-party application run its code at once. Apple states that this comes at the cost of running down the battery. Some disagree, however, saying that properly designed programs with sleeping processes remain dormant until woken, consuming no CPU resources, hence no battery drain.

Because of its fears about battery consumption, Apple has a strict "one application may run at a time" policy. Instead, push sends computing off-device. Performing work externally spares local CPU cycles and lets these external services produce asynchronous alerts without letting more than one third party program run at a time.

Some scaling required

The problem with push, at least from a developer point of view, is scaling. Background computing doesn't need to scale. A backgroundable application might involve an investment in coding, figuring out how to let an application do its work without presenting a user interface, to sleep when it's not active, and to generally be a good citizen. But when that coding is done, the app runs on each device using the local CPU. Should a developer deploy an extra 10,000 copies, there's no further investment involved other than increased technical support.

Push computing does scale. Whether you have 10,000 or 100,000 or 1,000,000 users matters. That's because developers must provide a service layer that handles the operations for every unit sold. The more users supported, the greater the costs will be. Consider that these services need to be completely reliable and that consumers will not be tolerant of extended downtimes.

For each user, a push-based service must provide the same service off-line that the user would normally have running on a phone itself, such as checking for new instant messages or looking for new stories on an RSS feed. As the number of users increases, the load that places on a server increases proportionately. To build a reliable fault tolerant server system like that, even with a smart cloud computing solution, puts a real burden on finances. It demands a serious hosting setup.

Consider an application with just 10,000 users. It might service a million uses per day, assuming update checks every 15 minutes. More time-critical uses might demand checks every few minutes or even several times a minute. As the computational burden builds, so do the hosting costs. While cloud computing provides an excellent match to these kinds of needs, that kind of solution comes with a real price in development, maintenance, and day-to-day operations.

On top of reliability, add in security concerns. Many polled services require secure credentials. Those credentials must be
uploaded to the service for remote use rather than being stored solely on the device. Even if the service in question does not use that kind of authentication, the unique identifier that allows a service to contact a specific phone is sensitive in itself. Should that identifier be stolen, it could let spammers send unsolicited alerts. Any developer who enters this arena must take these possible threats seriously and provide highly secure solutions for storing and protecting information.

These concerns, when taken together, point to the fact that push notifications are serious business. It's likely that most small- and medium-sized development houses may completely opt out of being push providers. Between infrastructure and security concerns, the infrastructure it will take to properly offer this kind of service may price itself out of reach for those developers. Background processes may not be welcome to Apple and other handset manufacturers, but when looked at closely they offer a more economically sound approach for developers.