Instant Messaging and Presence: A Cautionary Tale

In one sense, instant messaging has been a part of the Internet toolkit from very early days, as many of the systems which were connected to the early network had multi-terminal interaction systems (like the TOPS-20 talk command). The extension of those into multi-system usage was a natural evolution, with the Unix version of talk gaining the facility in 1983, spawning a set of further extensions (ytalk, utalk, etc.) Tests for a user's presence were similarly early, with Name/Finger being described in 1977. With those early antecedents, we might expect that instant messaging and presence would be part of the interoperable core of the Internet protocol suite, well-defined and deployed nearly universally.

The history is a bit different. Those early systems have been replaced multiple times, first by IRC and then by a series of different efforts, many using proprietary protocols and organization-specific presence registration. One aspect of that history is that many large service providers had internal IM systems that they opened to the Internet; AOL's AIM, Microsoft's Windows Messenger, and Yahoo! Messenger are examples of this. They did not share underlying protocols or formats and so were not part of a larger, single system.

The IETF attempted to address that fragmentation in the IMPP working group whose BoF proceedings demonstrate a will to create an interoperable system but a bit of a bake-off problem: SIP, TOC, and Zephyr were each proposed. While it was eventually chartered to "define protocols and data formats necessary to build an internet-scale end-user presence awareness, notification and instant messaging system.", the group instead created a model and two data formats: one for message exchange among systems and one for presence information. The protocol work, normally the core of an interoperability effort at the IETF, simply dropped away.

The reason for that was a practical one: several of the largest existing IM systems were completely uninterested in adopting any new protocols. The existing systems provided a lock-in to their services and ensured that the network effects of a large user base remained in their own hands. As was clear in the proceedings of IETF 48, even the protocols being developed within the IETF context (like SIP) were reluctant to use other protocols for any key features. Without a willingness to adopt new protocols, large sets of existing users would be outside any new system dependent on them. No matter how well-built the architecture, the resulting system would have a serious problem building up a network effect strong enough to counter those already locked into the existing systems.

As a result, the group shifted to working out how the different systems could exchange messages and presence data instead. Though these might never be the common internal representations, the idea was that it would allow any system interested in exchanging messages outside its own garden to implement a single format and method to interchange with any other system. Given the reluctance of the major parties to adopt new protocols, this seemed like a reasonable interoperability strategy to propose. There was even a first customer, as the SIMPLE and XMPP camps looked willing to adopt it for interchange.

Despite that willingness, the resulting mechanisms saw very little use. While some of that was based on the ongoing reluctance of the existing providers, a good bit of it is actually was because a completely different interoperability strategy took off: the multiservice client. Originally called GAIM to signal its ability to work with AOL's AIM, the first of these clients added the ability to work with Microsoft, Yahoo Messenger, and a host of other niche systems. Instead of the backend systems interchanging messages on behalf of the user, in other words, the users interacted with each system for themselves.

These new clients didn't create a single identity for the user but instead logged into each of the systems with the user's existing credentials. While that had the downside of creating a problem in tracking who was on which system, it also allowed individuals to use different identities and to maintain different circles with each. If a user had a work account on Windows Messenger, for example, messages to friends on AIM did not have to pass through Microsoft via CPIM and PDIF; the user's multiservice client simply logged into both. That improved latency as well as providing some level of isolation.

It was good enough that the systems around during IMPP's tenure did not feel any great user pressure to adopt CPIM or PDIF, and they failed to take root. Had the multiservice clients never emerged, IMPP might have eventually succeeded, but some caution in making that conclusion is warranted. Creating protocol mechanisms that encouraged a specific interoperability strategy was the limit of the IETF's power here; it could smooth the way, but no more. In this case, it wasn't enough.

As we consider the more general question of how design expectations and deployment realities diverge, it's important to recognize that the interoperability strategy of the implementers is a key factor. If they are uninterested in interoperation at all, efforts to produce new standards will be a waste of time. Even when they are interested in some level of interoperability, the initial design expectations can be overcome by events as other strategies come into play or other players enter the field. In those cases, any protocol features which were built with a specific architecture in mind may be abandoned or may need to change. For protocol designers, that signals a need to consider carefully both how small the building blocks to be developed should be and how likely they are to be reused rather than remade.