On Helicopters and Submarines

Marshall T. Rose, Invisible Worlds

Bernoulli vs. Archimedes—Whenever you see a movie that's got a vehicle that's part helicopter and part submarine, you know you're in for a real treat. What could be cooler? One second, the hero's being pursued by some fighter jets piloted by some nasty dudes with bad haircuts, dodging air-to-air missiles and exchanging witty repartee over the radio with a megalomaniac bent on world domination; and then, just as the hero is unable to evade the very last missile, he pushes a button, the craft dives into the ocean, and is surrounded by an oasis of peaceful blue.

The sad part is that no one is going to be building one of these sweet babies anytime soon. As noted in *Intuitor*, a Web site that analyzes "insultingly stupid movie physics":

> Design parameters for helicopters are almost complete opposites of design parameters for submarines. Helicopters require lightweight construction to be able to fly. Subs need heavyweight construction to sink. Helicopters rise using Bernoulli's principle while subs rise primarily using Archimedes' principle. Subs must resist extreme external pressure while helicopters do not need to, etc., etc. There is virtually no way to design such a dual-purpose craft, nor is it likely there will be one anytime in the foreseeable future. (http://www.intuitor.com/moviephysics/AI.html)

Helicopters are great, and so are submarines. The problem is that if you try to build one vehicle to perform two fundamentally different jobs, you're going to get a vehicle that does neither job well.

What does any of this have to do with instant messaging (IM)? Well, the Session Initiation Protocol (SIP) is an excellent helicopter, but it is also being proposed for use as an instant messaging submarine. The proposal is known by a clever acronym, SIMPLE (SIP for instant messaging and presence leveraging extensions), but the SIP/IM approach doesn't have any of the good features normally associated with simplicity.

SIP does a great job AS A HELICOPTER, BUT WHEN YOU TRY TO MAKE IT FUNCTION AS AN IM SUBMARINE AS WELL, DISASTER MAY FOLLOW.

INSTANT MESSAGING AND SIP

SIP is a rendezvous protocol used to establish media streams (e.g., voice over IP, conferencing, and so on). The key thing to understand about rendezvous protocols is that they play an important but very limited role in data communications. They negotiate all the parameters necessary for data exchange to occur; but their role is also limited, because once this negotiation completes, the rendezvous protocol goes away and the actual exchange of data occurs.

Like all good protocols, SIP's design parameters reflect its operating environment. What this means is that SIP's design isn't optimal for use in other scenarios. For example, because the rendezvous protocol is used for brief exchanges, and comprises such a small part of an overall mix of data traffic (in comparison to the actual data exchange), SIP doesn't need to have a congestion-sensitive transmission algorithm. After all, SIP is trying to do only one or two handshakes, so using something like slow-start is actually counterproductive.

The difficulty here is the same thing that afflicts most protocols that achieve cult-like popularity: SIP is being considered for use in all kinds of different applications. (In fact, the magnitude of requests for SIP extensions has reached the point where there's actually an evolving review process for SIP modifications.) Regardless, inasmuch as a given extension is consistent with SIP's design parameters, the extension is probably a good thing. Similarly, if an extension is contrary to those design parameters, then the extension is a bad idea.

SIP/IM

There are two ways to do SIP/IM: the paging model and the session model.

The paging model. The paging model works by piggybacking instant messages in SIP's payloads. All of the data exchange takes place during the rendezvous negotiation. If all we're talking about is a couple of IMs here and there, then this is no big deal. However, such a mechanism simply isn't scalable.

Here's why: SIP, by default, uses the User Datagram Protocol (UDP) as its transport, and congestion-sensitivity is not available in SIP. After all, rendezvous protocols don't need it (but data exchange protocols do). What this means is that as the IMs start going back and forth, none is flow controlled—so SIP/IM traffic is just like PointCast traffic, RealAudio traffic, and so on.

To its credit, the piggybacking proposal makes some recommendations with respect to congestion control (e.g., SIP/IM should try to use TCP instead of UDP, wherever possible). However, since the rest of the SIP infrastructure is using UDP, there's no way to know if TCP is even possible, and there's no way to force a SIP intermediary to use TCP (even if it were available).

This is exactly the kind of advice that will never get taken in real life.

Although you can get the paging model to work in low-volume environments, as soon as you have meaningful IM traffic, all traffic starts to suffer. Among other things, scalability requires congestion control. Therefore, applications such as SIP/IM's paging model, which isn't congestion-sensitive, are simply bad network citizens.

To make matters even worse, because each IM is carried in a new SIP rendezvous, each time an IM is sent, a new SIP handshake is made. So, each IM generates two or more distinct packets instead of one. Think of it this way: Call setup is a lot more expensive than data transfer, but since you're supposed to do it only once per session, that's OK. Call setup is also a fixed cost, which is supposed to be a lot smaller than the variable cost of the data exchange that follows. But, what's not okay is when each session contains exactly one message, because that's when the fixed cost dominates the session—but, of course, that's exactly what the paging model of SIP/IM gives you. (So it has all of the overhead of TCP's connection setup, with none of the benefits of a stateful session.)

Now, is any of this SIP's fault? No, of course not. SIP is designed as a rendezvous protocol, not a data-exchange protocol. The paging model of SIP/IM is simply contrary to SIP's intended usage. To mitigate this mismatch, the SIP/IM folks claim that the paging model will be used only in limited, special-purpose environments (e.g., when talking to cellphones). This really isn't very helpful, since there isn't any way to know which mode is appropriate in a given context, because the remote peer is, by definition, remote. Again, this is exactly the kind of advice that will never get taken in real life.

The session model. The session model works by using SIP to negotiate the parameters for an IM stream and then using some other protocol to exchange the actual instant messages. Hey! This approach actually makes sense. Oh, wait. The group working on the session model hasn't developed a protocol to do that yet. Oh well, maybe later.

IF THE ONLY TOOL YOU HAVE IS A CHAINSAW...

...then everything starts to look like a tree. Regardless of whether you choose the paging or session model for SIP/IM, there are still two serious issues to be concerned about.

Rendezvous is not routing. *Rendezvous* is about finding common ground for interaction. That's what SIP does. *Routing* is about finding a path between end-points. That's what the Routing Information Protocol (RIP) and Border Gateway Protocol (BGP) do, and what IM needs. This distinction explains why doing IM over SIP isn't as "obvious" as it might seem.

Recall that SIP is designed to solve the rendezvous problem for different applications. As such, it shouldn't be surprising that SIP has tons of stuff in it that isn't needed to do IM. For example, in the basic SIP model, you have six (yes, six!) interaction modes for four entities (clients, servers, proxy servers, and redirection servers). In most IM systems, you have (at most) two interaction modes involving (at most) two kinds of entities. So the poor implementer of IM's submarine functions first has to build a nice, shiny helicopter.

You're not going to get any savings through integrating IM with your SIP infrastructure.

The bottom line, of course, is that deploying SIP/IM has a much higher cost than other IM approaches, because deploying SIP has a high cost. This is actually the sly humor in the SIMPLE acronym for SIP/IM: Once you've deployed SIP, you simply won't notice the incremental expense of deploying IM. That's just great, assuming, of course, that you needed SIP to begin with. If not, well, just think of it as a rather regressive

tax on your IM deployment.

Even when—or if—SIP becomes universal, the claim that the incremental cost is small is not entirely fair. You see, SIP does rendezvous, but what IM really needs is routing—and rendezvous and routing, while similar, are not the same. In other words, even if you do need SIP in your part of the Internet, adding IM on top of SIP isn't free—you still need to deploy an IM-based set of rules on top of that. You're not going to get any savings through integrating IM with your SIP infrastructure.

Here's a second example of why rendezvous isn't routing: When many users in two enterprises communicate—say, via e-mail—the mail routing infrastructure is able to bundle up the traffic together (e.g., a single connection is typically established that handles multiple message flows). Rendezvous systems don't have this concept—or efficiency of aggregation.

Feature interference. One of the reasons the powers-that-be had to create a review board for SIP extensions is to understand the risk of feature interference. When SIP gets used for multiple applications, each application may have its own set of extensions, and extensions have to be implemented not only on end-user systems but also on SIP intermediaries.

Then, it's possible—or inevitable—that the extensions are going to interfere with each other.

Of course, this isn't unheard of. Consider HTTP: Whenever people propose implementing some service over HTTP, they also have to worry about feature interference—that is, caches and proxies will do things that are just fine for services based on static HTML pages, but are counterproductive for the new service.

The real question is whether you want your IM system to be subject to that kind of risk. Reuse is a great thing—as long as the things you're reusing don't interfere with the original job you're trying to get done.

WITH ENOUGH TRUST...

...even a cow can achieve orbit. The question, of course, is whether putting a cow into orbit is a potentially worthwhile endeavor (with apologies to Monty Python).

The 1999 version of SIP is 153 pages long, but as of this writing:

- The current version is more than 350 pages long (comprised of RFCs 3261-3265).
- A half-dozen or so SIP extensions are in the RFC editor's queue slated for the standards track.
- Another half-dozen SIP extensions are in the Internet Engineering Steering Group's (IESG's) queue slated for the standards track.

None of these numbers includes any of the individual SIP submissions, informational or experimental SIP specifications, or IM-specific SIP specifications. Further, none of these numbers includes anything currently being developed in any existing working group.

The message here is that SIP is a large work in progress. Of course, there's nothing wrong with that: Lots of folks want to use SIP to do things, and to the extent that the things they want to do are consistent with SIP, that's great! If you're trying to deploy an IM system, however, then the wait for "ubiquitous SIP" might have an impact on your decision.

THE POINT OF ALL THIS

If you need to negotiate rendezvous parameters before establishing a media stream, use SIP. SIP is not the problem.

Rendezvous protocols are great, and so are data-exchange protocols. The problem is that if you try to build one protocol to perform two fundamentally different jobs, you're going to get a protocol that does neither job well. In other words, SIP and IM are sufficiently different that trying to do them both in the same protocol is problematic. Q

LOVE IT, HATE IT? LET US KNOW:

feedback@acmqueue.com or www.acmqueue.com/forums

MARSHALL T. ROSE, Invisible Worlds cofounder, CTO, and general manager, is the prime mover of the BEEP Protocol and the former CTO of MessageMedia, a leading source of e-messaging services. Marshall formerly held the position of the Internet Engineering Task Force (IETF) area director for network management, and was one of a dozen individuals who oversaw the Internet's standardization process.

Marshall has been responsible for the design, specification, and implementation of several Internet-standard technologies and is the author of more than 60 of the Internet's requests for comments (RFCs). He is the author of several professional texts on subjects such as Internet management, e-mail, and directory services. Rose is well known for his implementations of core Internet technologies (such as POP, SMTP, and SNMP) and OSI technologies (such as X.500 and FTAM). In 1981, Rose worked on his first openly available large-scale project, the MH mail system. Rose received a Ph.D. in information and computer science from the University of California, Irvine, in 1984.