In the finest tradition of software project management, the Payroll development team is focused on processing the employee database records against the latest tax regulations, while their counterpart in Accounts Payable is reading those reports into its database to issue direct-deposit payments to employees' banks. Interconnecting the two processes is a separate integration step. But how to proceed?

In the previous release of this payroll and accounting software, which ran on a single mainframe, the two processes could communicate only via shared batch files. Today there are more choices: sharing a distributed file system, e-mailing the check records, pulling the records off a Web server, or writing yet another custom message-transfer protocol.

However, each choice requires trade-offs involving latency, reliability, security, and a hundred other details. That is why we have so many Internet-scale messaging systems: Telnet for duplex byte-streams, File Transfer Protocol for pulling files, Simple Mail Transfer Protocol for reliable e-mail messaging, Network News Transfer Protocol for pushing e-mail, HyperText Transfer Protocol for pulling Web pages, and a host of other special-purpose schemes, from Finger to Network File System.

SORTING THEM OUT

"Seventh Heaven" is a column about these choices at the seventh layer of the Open Systems Interconnection stack (see Figure 1). Application-layer protocols proliferate in myriad forms every time a developer wires together software components across the Internet. The Internet Engineering Task Force's applications area is growing rapidly, in contrast to the relative stability of TCP/IP. Bookstore shelves groan under the weight of new volumes, each cataloging a different species of transfer protocol that evolved in response to the vagaries of network topology, performance, economics, and standards-body politics. How can we make sense of it all?

Through this column, I will develop a framework to help navigate among the alternatives. Each subsequent installment will examine one of the TPs in context. Understanding their history is key to charting their future, especially as new contenders like HTTP-NG (next generation) emerge. This first installment sets forth our general ontology—a vocabulary for describing and classifying TPs.

LIFE IN A TP

At lower levels of the network stack, designers deal with transport protocols like TCP and TP4; at the application layer, they work with message transfer protocols. The latter consist of three major components:

- addresses, to identify participating nodes,
- distribution rules controlling transfers, and
- the messages themselves.

Admittedly, the framework I outline here is not explicit in the design rationale of any TP to date—requests for comments (RFCs) emphasize how the protocols work almost to the exclusion of why—but I believe the synthesis will be useful going forward.

TP characteristics such as reliability, latency, and maintainability can be explained as consequences of variations along the three axes of addressing, distribution, and content, shown in Table 1. E-mail, for example, has reliable handoffs at each step, but does not establish a direct mailbox-to-mailbox connection. Thus, actual delivery can take a while, usually without delivery notification.

Furthermore, each handoff is traceable through Received: headers for debugging. In contrast, HTTP is a synchronous request-reply protocol that requires direct, online connections. With HTTP, you can directly compose transport-layer security to encrypt the end-to-end connection; with e-mail's hop-by-hop transfers you can only protect communication between mail servers, not end users.

NAMES AND ADDRESSES

The first things a TP must define are the addresses of the nodes it transfers...
messages between and the names for the entities it transfers.

For many TPs, the endpoints have network interfaces on the connected Internet, so they use domain names or IP addresses. Sometimes the endpoints are logical concepts, such as individual users’ mailboxes or globally distributed newsgroups. The entities they purport to transfer must be represented as messages, so the entities’ names must identify the message at hand. These names are more intimately tied to the semantics of the service: Telnet provides a Network Virtual Terminal to some port on a host; FTP and HTTP servers can be queried for the value of particular pathnames on a host; and SMTP and NNTP refer to messages by their globally unique RFC-822 msgids.

There may be additional relations between names and addresses. An HTTP caching proxy uses names that include the original host address (for example, http://cache/http://.../...). Similarly, an FTP mirror site maintains a copy of a file at a different host, possibly with a derived name (containing, for example, a prefix like /mirrors/sumex-aim/...). However, once fetched, a file’s name is the local file system pathname; FTP does not use metadata to bind it to its original name.

PUSH AND PULL
Traditionally, a TP’s different distribution rules are its most salient feature for classification. When we speak of distribution rules, however, we are speaking of two levels of description. At a mechanical level, the TPs are built atop TCP, so it is natural for clients to initiate the connection. However, in deployed applications we can speak of a more abstract intent: for senders to push data at their will, or for receivers to pull data at theirs. Traditional Web clients can only fetch data from servers, but FTP service usually allows uploading and downloading, so the net effect can be push or pull (although FTP service built into Web browsers cannot upload, unfortunately).

Another significant aspect of distribution rules is their topologies. Again, while TCP/IP is a point-to-point service, the net effect of message delivery can vary. Telnet is strictly a one-on-one service. On the other hand, e-mail, with its intermediate relays and envelopes with multiple names, can provide broadcast messaging to a set of mailboxes. Similarly, although HTTP connections are one-on-one, publishing a URL amounts to broadcasting the data. Multicasting experiments, such as Newscaster (http://user.cs.tu-berlin.de/~nilss/som/newscast.htm) and Self-Organizing Multicast Transfer Protocol, are changing these rules, however, by offering multipart messaging.

Finally, choosing synchronous or asynchronous messaging determines whether and how TPs can be represented by other parties. Unlike simple tunneling, proxy service implies that a message is reprocessed by an intermediate node. Synchronous conversations, such as HTTP, can only be chained, while asynchronous handoffs can allow an asymmetric path, as in e-mail routing. Synchronous one-to-many broadcasting can also cause “ACK implosions,” which explains why real-time broadcasting is usually designed to accommodate loss rather than process acknowledgments.

**MEATDATA** AND METADATA
Messages are why we design TPs in the first place. Most TPs are designed with a particular type of content in mind and optimized to that end. Moreover, the elemental protocol data units in these systems are messages that combine an envelope, the command and metadata about the entity, and a concrete representation of the entity itself.

Message content affects the choice of transport in TPs: login emulation with Telnet requires the URGent delivery flag in TCP segments; delivery of open-ended streaming data motivates a separate TCP connection for each HTTP/1.x operation; and time-critical multimedia content may tap User Datagram Protocol datagrams directly.

The bytes that constitute a message usually combine the contents and the distribution algorithm’s commands (FTP, however, separates its control and data channels). In other words, messages combine metadata—some snapshot of the resource itself—and metadata. Typically, the source and destination addresses and transaction logs describe the command; content-type, content-name, and content-lifetime information (for caching) describe the entity. Traditional remote procedure call (RPC) and distributed-object protocols also exchange lightweight messages, but to date they have not had caching and reflection. Furthermore, advanced TPs can also use metadata to optimize distribution, as with HTTP’s content-negotiation and cache-validating HEAD requests.

**ASCENDING TO SEVENTH HEAVEN**
Application-layer developers, like those for Payroll and Accounts Payable, have a new pivot point for deciding how to transfer messages between software modules. The classical single-necked model of Internet development—com-
mon interoperability of IP packets—is becoming a double-necked model, as shown in Figure 2. We have application-layer consensus around Uniform Resource Identifiers for naming and addressing, and MIME-like messages for envelopes and data.

Sir Isaac Newton may have “seen further by standing on the shoulders of Giants,” but as Digital Equipment Corporation researcher Brian Reid quipped, “In computer science, we stand on each other’s feet.”

What have we learned about the evolution of TPs over the past 15 years? Consider the mailing list of an electronic community I set up: Each posting is delivered as e-mail, as a posting in a local newsgroup, as an archived Web page (both at my own site and at Findmail.com, resulting in different URL paths), and as FTP’able mailbox files. This is certainly not an unusual arrangement, so why do we require the same resource—words in an ongoing conversation—to adopt a half-dozen different names, addresses, message formats, and distribution rules to reach different people?

Why isn’t there a stable identifier and storage format that not only bridges them all but also accommodates future communication modalities? I mean, just imagine integrating ephemeral references to a real-time video chat room!

In Jewish mysticism, the first level of heaven is the sky and clouds, then the planets, and on up through succeeding levels to the stars and finally the ultimate nature of the Divine (see http://www.acs.ucalgary.ca/~elsegal/Shokel/891103_7th_Heaven.html). “Seventh Heaven” also aims at understanding the truth of the ultimate layer.

If you have any secret seals bearing the true name of the Divine, or just comments on this taxonomy, please feel free to contact me.