The Technical Development of Internet Email

Craig Partridge
BBN Technologies

Development and evolution of the technologies and standards for Internet email took more than 20 years, and arguably is still under way. The protocols to move email between systems and the rules for formatting messages have evolved, and been largely replaced at least once. This article traces that evolution, with a focus on why things look as they do today.

The explosive development of networked electronic mail (email) has been one of the major technical and sociological developments of the past 40 years. A number of authors have already looked at the development of email from various perspectives.[1] The goal of this article is to explore a perspective that, surprisingly, has not been thoroughly examined: namely, how the details of the technology that implements email in the Internet have evolved.

This is a detailed history of email's plumbing. One might imagine, therefore, that it is only of interest to a plumber. It turns out, however, that much of how email has evolved has depended on seemingly obscure decisions. Writing this article has been a reminder of how little decisions have big consequences, and I have sought to highlight those decisions in the narrative.

Architecture of email

In telling the story of how email came to look as it does today, we start by describing (in broad strokes) today's world, so that the steps in the evolution can be marked more clearly.

Today's email system can be divided into two distinct subsystems. One subsystem, the message handling system (MHS), is responsible for moving email messages from sending users to receiving users, and is built on a set of servers called message transfer agents (MTAs). The other subsystem, which we will call the user agent (UA), works with the user to receive, manage (e.g., delete, archive, or print), and create email messages, and interacts with the MHS to cause messages to be delivered. Readers may recognize this terminology as being roughly that developed by the X.400 email standardization process.

Each subsystem internally has a rich set of protocols and services to perform its job. For instance, the UA typically includes network protocols to manage mailboxes kept on remote storage at a user's Internet service provider or place of work. The MHS includes protocols to reliably move email messages from one MTA to another, and to determine how to route a message through the MTAs to its recipients.

The UA and MHS must also have some standards in common. In particular, they need to agree on the format of email messages and the format of the metadata (the so-called envelope) that accompanies each message on its path through the network.

The focus of this article is how these different pieces incrementally came into being and exploring why each one emerged and how its emergence affected the larger email system. In the interests of space, this survey stops around the end of 1991. That termination date leaves out at least four stories: (1) the development of graphics-based user interfaces for personal computers and the incorporation of those interfaces into web browsers; (2) the rise of UA protocols such as the Post Office Protocol (POP)[2] and IMAP[3] (these protocols existed prior to 1991, but much of their evolution occurred later); (3) the continuing efforts to further internationalize email (e.g., allowing non-ASCI characters in email addresses); and (4) the rise of unwanted email (dubbed ''spam'') and tools that sought to diminish it. Furthermore, in the interests of space, I do not consider the development of technical standards for the support of email lists.

First steps

Electronic mail existed before networks did. In the 1960s, time-shared operating systems

developed local email systems delivering mail between users on a single system.[4] The importance of this work is that email requires a certain amount of local infrastructure. There needs to be a place to put each user's email. There needs to be a way for a user to discover that he or she has new email. By the early 1970s, many operating systems had these facilities.

In July 1971, Dick Watson of SRI International published an Internet Request for Comments[5] (RFC-196) describing what he called "A Mail Box Protocol." The idea was to provide a mechanism where the new Network Information Center (NIC) could distributed documents to sites on the Arpanet. Watson described a way to send files (documents) to a teletype printer, with different mailboxes for different types of printers. Mailbox 0 was a teletype

> assumed to have a print line 72 characters wide, and a page of 66 lines. The new line convention will be carriage return (X'0D') followed by line feed (X'0A') … The standard printer will accept form feed (X'0C') as meaning move paper to the top of a new page.[6]

Ray Tomlinson of Bolt Beranek and Newman (now BBN Technologies or BBN) read Watson's memo and reacted that "it was overly complicated because it tried to deal with printing ink on paper with a line printer and delivered the paper to numbered mailboxes."[7] In Tomlinson's view, the correct approach was to send documents to a user's electronic mailbox and let the user decide if the document merited printing.[8] So Tomlinson set out to see if he could send email this way between two TENEX systems[9] over the Arpanet. His approach was simple.

TENEX already had an existing local email program called SNDMSG,[10] which, given a message, appended that message to a file called MAILBOX in a user's directory. TENEX also had a homegrown file transfer service called CPYnet (written by Tomlinson). In a passive mode, CPYnet listened at a particular address for requests to read, write, or append to a particular local file. Email was achieved by incorporating CPYnet into SNDMSG. If SNDMSG was given a message addressed to a user at a remote host, it opened a CPYnet connection to the remote host and instructed CPYnet to append the message to the user's mailbox on that host.

Users learned that they had received network email the same way they learned they had received local email. In TENEX, they got a "You have mail" message when they logged in. Mail was read by viewing or printing the mailbox file, usually with the TYPE command. (Almost immediately, TYPE MAILBOX was replaced with a TENEX macro READMAIL). Messages were deleted by deleting the relevant lines with a text editor.

Tomlinson made two important contributions. First, he found a way to express the networked email address. He chose to use the "@" sign to divide the user's account name from the name of the host where the account resided, resulting in the now ubiquitous *user@remote* format.[11] Second, SNDMSG was the first MTA—it took a message and delivered it (using the CPYnet protocol) to a remote user's mailbox.

Observe that the last contribution is a surprise. We might imagine that the first program was more of a user agent (UA) than a message transfer agent (MTA). But SNDMSG could only deliver mail, it could not receive mail, and it delivered the email all the way to the recipient's mailbox. Therefore, SNDMSG was much closer in spirit to an MTA (and, indeed, as we shall see, was used as an MTA for a number of years). At the same time, SNDMSG was primitive. If there were multiple email recipients on the same host, it copied the message once for each recipient. If the remote host was down, SNDMSG simply returned a failure message—it made no effort to retransmit.

Despite its primitive nature, Tomlinson's creation took off. The next few years saw it mature from a fun idea to a central feature of the Arpanet (and later the Internet).

From primitive to production

By late 1973, email was widely used on the Arpanet. What happened after Tomlinson's experiment to make this happen? Obviously, email met a need. But there were also technical steps: standardization of the transfer protocol and the development of user interfaces.

A standard transfer protocol

First, the community replaced CPYnet with a standardized file transfer service, the first generation of the File Transfer Protocol (FTP). This process took a while. In 1971, FTP was simply a set of rather complex ideas written up in a set of RFCs by a team led by Abhay Bhushan of the Massachusetts Institute of Technology (MIT).[12] The goal behind these ideas was to create a general tool to manage files (including deleting and renaming files) on

remote machines and to do it in a way that met the needs of any envisioned application.[13]

At the same time, Dick Watson's mailbox idea was continuing to mature. In November 1971, a team including Watson proposed a way to enhance (the still nascent) FTP with an explicit MAIL command to support appending a file to a mailbox. They further proposed that email be simply ASCII strings of text (no binary images) and that mailbox numbers be replaced with text user identifiers. The identifiers were ''NIC handles.'' NIC handles were given out by the Network Information Center to authorized network users (and were used as login IDs on Arpanet terminal servers, called TIPS). This idea, of course, meant that every host would need to maintain a table mapping NIC handles of local users to the location of their mailbox file. Retaining Watson's original idea of accessing a printer, the MAIL command could be given the name ''Printer'' instead of a NIC handle and the file would be printed.

Concurrently, Tomlinson distributed SNDMSG to other TENEX systems and people began to get hands-on experience with email. TENEX was the most common operating system on the Arpanet at the time, and so probably at least half the Arpanet users had access to SNDMSG.

In April 1972, most of the interested parties, including both Tomlinson and Watson, met at MIT to discuss revisions to the File Transfer Protocol. The meeting made several decisions, at least one of which proved to have a long-term impact: the group agreed to use text (ASCII) commands and replies (previous versions of FTP had used binary commands) to aid interactive use.[14] To this day, the Internet uses text commands to transfer email (and the tradition lives on in much later protocols, such as the Web's transfer protocol, HTTP). A new version of the FTP specification, based on these ideas and written by Bhushan, came out in July 1972.[15]

The new specification envisioned that email would be delivered via the APPEND command, which appended data to a file. Discussions about FTP and email continued, however, and a month later, Bhushan issued a revision to the FTP specification[16] to include a new command, MLFL (Mail File). It is said Bhushan came up with MLFL because, one evening while he was writing the revision, a fellow graduate student at MIT stopped by to suggest that a better solution was required for email.[17]

MLFL took one argument, a user id, which could either be a NIC handle or a local user name (local to the remote host). The user id could also be left out, in which case the mail was to be delivered to a printer. After the MLFL command was accepted, the email file was transmitted over an FTP data channel (with the end of the file indicating the end of the message). The file was required to be in ASCII. A separate copy of the file was sent for each recipient at a host.

MLFL was an important step. A key flaw in Tomlinson's prototype email was that you had to know where in the receiving host's file system a user's mailbox was located, so that you could append to it.[18] This limitation probably explains why most of the email activity in 1971 and 1972 appears to have taken place between TENEX systems, where the file name for the mailbox was consistent. MLFL adopted Watson's notion that mailboxes are symbolic names that the receiving system translates into an appropriate user mailbox file and thereby freed email from system-specific limitations.

An interactive command, MAIL, was also defined, so that users logged into a TIP could type in an email message using only FTP's control connection. In this case, a line with a single dot (''.'') on it marked the end of the message. Ending a message with a single dot is still how email is moved over the Internet today.

The MAIL—and, more important, MLFL—commands remained the way email was delivered between systems for several years.

In the fall of 1972, Bob Clements of BBN updated SNDMSG to use the new commands. Several other email-cognizant FTP implementations appeared. The most notable is probably the system for MIT's Multics. Ken Pogran wrote the FTP implementation and Mike Padlipsky wrote the NETML program that handled email.[19] Multics was exceptional for the time because it had good security including user file privileges, so Padlipsky had to invent a special user (ANONYMOUS) to receive email and distribute it to users.[20] The concept of an anonymous login account caught on as a way to permit FTP access to users who did not have an account and remains a central feature of FTP to this day.

First user agents

The second development of 1972 and 1973 was the creation of tools to create and manage email. Here the center of innovation was within the Advanced Research Projects Agency (ARPA) itself. Larry Roberts, head of the ARPA office funding Arpanet, was an early and aggressive user of email. Early in 1972, Stephen

Lukasik, the head of ARPA, also began using email and that induced a number of others, including the ARPA department heads, to use email too.[21]

Soon Lukasik became frustrated with READMAIL, which forced him to read through all the messages in his mailbox in order. Lukasik liked to keep copies of email he received, which made the problem worse. He appealed to Roberts for something better.

One night in July, Roberts wrote a tool using macros for the TECO (Text Editor and COrrector[22]) text editor to manage a mailbox.[23] The tool was dubbed RD. RD made it possible to list the messages in the mailbox, to pick which message to read next, and to print individual messages.

Roberts' colleague at ARPA, Barry Wessler, promptly rewrote RD as a standalone program in the programming language SAIL and added additional features for usability. Improvements in Wessler's ''New RD'' or NRD included the ability to manage more than one file of messages, and mechanisms to file, retrieve, and delete messages. RD and NRD were the first mailbox management tools, the first true user agents.

Wessler's NRD was not distributed outside ARPA. (RD was.) In early 1973, Martin Yonke was a graduate student intern at the University of Southern California's Information Sciences Institute (ISI) and looking for something to do. Steve Crocker of ARPA gave Yonke a copy of Wessler's code (which ran on TENEX) and suggested Yonke look at improving it. Yonke added command completion (type the first letter or two of a command and the rest of the name would be filled in) and a help interface. A user could type a question mark in most places in a command to learn what the choices were. The revised NRD was dubbed BANANARD.[24] (At the time, ''banana'' was technical slang for ''cool'' or ''better''.) Yonke distributed and maintained BANANARD for a bit less than a year although it remained in use for several years more.

Among the amusing stories from that year, one concerned mailbox sizes: BANANARD kept an index of messages in a file, so Yonke had to estimate how big the index (which was read into memory) might be. Yonke estimated the largest possible mailbox size, doubled that, and concluded that assuming a mailbox was never larger than 5,000 messages was safe. Within a few months, Steve Crocker exceeded the limit. So did John Vittal.[25]

One challenge in RD and NRD was the lack of a standard format for email messages. Headers varied. It was hard to find where one message ended and the next one started. Wessler remembers trying to get NRD to find the start of headers, but it was too hard because messages routinely had other messages embedded in them. Therefore, NRD (and RD and BANANARD) relied on the receiving system to place a start-of-message delimiter before each message in the mailbox.[26] The delimiter had four SOH (Start Of Header, also known as Control-A) bytes followed by information about the message (initially just a byte count, later somewhat more information).[27] In one of those odd quirks, part of the start-of-message delimiter has lived on. While some present-day email systems parse for a header, others still expect messages separated by a line with four consecutive SOH bytes.

One challenge in RD and NRD was the lack of a standard format for email messages. Headers varied. It was hard to find where one message ended and the next one started.

Transitions

In March 1973, another meeting of people working on FTP was held, to try to clarify issues lingering from the April 1972 meeting. It marked a subtle transition.

Originally, clarifying and improving the support for email in FTP was part of the agenda.[28] Yet the meeting was ambivalent about the relationship between FTP and email. Prodded by a late-in-the-meeting arrival of ARPA's Steve Crocker, who asked how they were doing on email support, the group decided to formally incorporate the MLFL and MAIL commands into the new specification[29] (recall that the commands had previously been in a separate addendum). Between the meeting and the issuances of the new FTP specification, it was decided that email should really be a separate, auxiliary protocol.[30] Email had become important (or complex) enough to merit distinction.

Second, the community was shifting. Although both meetings had over 20 attendees, they were different sets of people. Only five people[31] attended both meetings.[32] Abhay Bhushan, who had been driving the development of and writing the specifications for FTP, would soon move on to other things. Nancy Neigus of BBN wrote the new FTP specification.

The research focus was also changing. By year's end, Larry Roberts (probably email's most important early adopter) would leave ARPA, and under his successor, Bob Kahn, ARPA's networking focus would change to developing networks over media other than telephone wires (e.g., satellites and radios) and the problems of interconnecting those networks.

Finally, at least from a standards perspective, the protocol for delivering email enters a kind of limbo. The auxiliary protocol specification for email envisioned in the new FTP specification never appeared. After three years, Jon Postel wrote a two-page memo that never appeared online, documenting the, by then well-established, practice of using MAIL and MLFL. The memo suggests some sites had not bothered to update their FTP from before the 1973 FTP meeting.[33] There were multiple attempts to allow FTP to send a single copy of a message to multiple recipients. All of them apparently failed.[34] It would take seven years from the FTP meeting before the community seriously returned to the problems of a new email protocol.[35] Innovation over the next few years would come from user agents and a long-running debate over the format of email messages, especially email headers.

Rise of the user agent

In early 1974, John Vittal worked in the office next door to Martin Yonke's office at ISI. Vittal had helped Yonke with BANANARD, and about the time Yonke stopped working on BANANARD so he could finish his graduate degree, Vittal took a copy of the code and began to think about building an improved user agent.

MSG

Vittal called his new program MSG. In it he sought to write a user agent that was simple yet did all the things a user needed it to do. It had roughly the same functionality as BANANARD, but the structure of its commands reflected feedback Vittal sought out from users about how they wanted to manage their email. MSG was a personal effort by Vittal (writing code on nights and weekends), and when he left ISI for BBN in 1976, he took MSG with him.

MSG was, in fact, surprisingly simple. It was a stand-alone program with its own set of commands. There were just 30 commands, named such that their first letter uniquely identified all but six. Combined with a command-completion scheme, this usually-unique-on-first letter approach permitted concise typing by experienced users. (Many early computer users were hunt-and-peck typists, so keeping commands to a letter or two in length was a big time-saver.)

Of these 30 commands, several were new from BANANARD. Some were minor, such as a command to toggle the user interface between a concise and a verbose mode. However, three commands reflect important changes:

- *Move* reflected Vittal's attention to user behavior. He noticed that one of the most common activities was to save a message in a file and then delete the message from the inbound mailbox. Vittal created the combined Save/Delete command, Move.
- *Answer* (now usually called ''reply'') is widely held to be Vittal's most insightful and important invention. Answer examined a received message to determine to whom a reply should be sent, then placed these addresses, along with a copy of the original SUBJECT field, in a responding message. Among the challenges Vittal had to solve were the varying email-addressing standards and what options to give a user (reply to everyone? reply only to the sender of the note?). It took three implementations to get right.[36]

  The wonder of Answer is that it suddenly made replying to email easy. Rather than manually copying the addresses, the user could just type Answer and Reply. Users at the time remember the creation of Answer as transforming—converting email from a system of receiving memos into a system for conversation. (There are anecdotal reports that email traffic grew sharply shortly after Answer appeared.[37])
- *Forward* provided the mechanism to send an email message to a person who was not already a recipient. How much of an innovation Forward was is unclear. Barry Wessler had to struggle with messages embedded in messages in NRD. But the formalization of the idea was new.

MSG became the Arpanet's most popular user agent and remained so for several years.

Hermes and MH

About the same time Vittal was starting work on MSG, Steve Walker at ARPA created a new committee called the ''Message Services Committee,'' charged with thinking about email issues. Its focus was on user agents (Al Vezza of MIT remembers a push to get user agents to support command completion) and email headers. In the summer of 1975, Walker also created the MsgGroup mailing list, to encourage greater discussion.[38]

Motivating these efforts was an ARPA program called the Military Message Experiment (MME) to make email into a useful service to the military. As part of this program, between 1975 and 1979, ISI, BBN, and MIT (in an advisory role) sought to create user agents designed for the needs of the military. The initial goal was a system for personnel at the office of the Navy Commander in Chief for the Pacific (CINCPAC).[39] In a related effort, RAND Corporation was funded to develop a Unix email user agent.[40]

Hermes (a BBN project) and MH (at RAND) were products of this program. Another system, called SIGMA, was developed by ISI for CINCPAC but never used elsewhere. They illustrate some of the diversity of user agents of the time. (An interesting side note is that John Vittal worked on both SIGMA and Hermes, while continuing his work on MSG. So Vittal's personal project was competing with the in-house official product. At both ISI and BBN, MSG won.)

Hermes was designed for an office (or command) environment where much of the email received was kept for reference. It contained a sophisticated set of mechanisms for filing and searching for messages, including a database that recorded key fields from each message to make searches fast. Hermes also provided a high degree of customization. Readers could create a template of how messages should be displayed, how they should be printed, and even how they should be created (what fields a user should be prompted for). To support this customization, Hermes had a per-user configuration file (called a profile) remembered as having been large and complex, though documentation suggests it was far simpler than the MH profile file became by the mid-1980s.[41] Initially known as the MAILSYS project, the Hermes team at various times included Jerry Burchfiel, Ted Meyer, Austin Henderson, Doug Dodds, Debbie Deutsch, Charlotte Mooers, and John Vittal.

MH (''Mail Handler'') was the successor and response to an earlier RAND system, called MS. MS was a user agent for the Unix operating system (apparently the first Unix user agent). MS was funded by Steve Walker at ARPA and was created by William Crosby, Steven Tepper, and Dave Crocker.[42] MS's defining characteristic appears to have been that it supported multiple user interfaces, including one that sought to mimic a Unix command shell and another that mimicked MSG.

Soon after MS was working in 1977, Stock Gaines and Norm Shapiro of RAND wrote an internal memo suggesting that MS was inconsistent with the style of other Unix programs.[43] Unix encouraged the use of many small programs, each of which did something well and creating metaprograms by combining the small programs together using a mechanism called ''pipes.''[44] Gaines and Shapiro suggested the same approach for email: a set of small programs that managed email, where email messages were stored as separate files in a user's directory.

Two years after the memo, a new RAND employee, Bruce Bordon, was assigned to upgrade MS. He recommended to his management that rather than upgrade MS, he should implement Gaines and Shapiro's idea. The result was MH.

The virtue of MH is that it makes email part of the user's larger environment.[45] Output of email display programs can be filtered through search programs such as *grep* or simply sent to the printing program. MH, in some ways anticipated today's world, where clicking on an attachment opens the correct program. Culturally, in Unix, rather than clicking on an attachment, one pipes data from one program to the next to produce the desired result.

Because MH puts every message in a separate file in a folder (directory), it is easy to manipulate both individual messages and folders. Accordingly, MH (unlike MS[46]) has powerful tools to sort folders and to search, mark, and label messages.

Through most of the 1980s, MH was maintained by Marshall Rose, with help from a number of people, most notably John Romine, Jerry Sweet, and Van Jacobson.[47] Others have picked up the task since and MH (much evolved in its code, but still recognizable as Bordon's suite of programs) continues to be widely used today.

Message formats and headers

When Ray Tomlinson sent his email between TENEX systems, he used a format similar to a business memo. But there was no standard format for email messages and creating and

revising standards for email message formats would consume a tremendous amount of effort over the next several years.

First message format standard

Abhay Bhushan, Ken Pogran, Ray Tomlinson, and Jim White (of SRI) took the first step to standardize email headers in RFC-561, published in September 1973.[48] Their proposal was mild. Every email message should have three fields (FROM, SUBJECT, and DATE) at the start. Additional fields were permitted, one per line, with each line starting with a single word (no spaces) followed by a colon (:). The end of this header section was marked by a single blank line, after which came the contents of the message.

The proposed standard was forward looking even as it lacked some basic features. The ability to make any word into a header field was progressive and left plenty of room for experimentation. The date field was surprisingly precise, specifying the time to the minute and the time zone. The blank line after the header remains a feature of email today. Yet there was no TO field, so a recipient wouldn't necessarily know who else was to receive the message, and, while use of the @ sign was already common, the address format required using the word "at," as in TOMLINSON AT BBN-TENEX, with the odd consequence that for several years, people would send emails using "at" in the FROM (and soon, TO) field and yet within the message itself list their email address with an "@."

Partial progress

In 1975, a team of people working on email systems at BBN sought to update RFC-561 with RFC-680.[49] The work was produced under the auspices of ARPA's Message Services Committee.[50] The RFC authors were Ted Meyer and Austin Henderson, but email on the MsgGroup mailing list suggests Charlotte Mooers[51] also played a major role. RFC-680 set out to document a large number of fields, many of which were already in widespread but informal use, and to standardize their formats in a way that computer programs (e.g., user agents) could easily parse.

That the header standard needed updating was becoming increasingly clear. Jack Haverty offered the following example from his time maintaining the MIT-ITS mailer.

> [A] field like "To: PDL, Cerf@ISIA" was ambiguous was "PDL" really "PDL@ISIA" (picking up the host from the end of the line)? Or was it "PDL@MIT-DMS" (picking up the host from the "From: JFH@MIT-DMS" elsewhere in the header)?
>
> Various mail programs adopted different such "abbreviations" which drove me crazy. … To handle all of this protocol chaos, I wrote (and rewrote, and tweaked) a sizable (for a LISPish world) chunk of code to try to deduce the precise meaning of each message header contents and semantics based on where the message came from. Different mail programs had different ideas about the interpretation of fields in the headers.
>
> That code first tried to figure out where an incoming message had come from. This was not so obvious as it might seem because of redistribution and forwarding of messages, and differences in behavior of various versions of the other guy's software. So it wasn't enough to just look to see if you were talking to MIT-MULTICS. I remember having conditional clauses that in essence said "If I see a pattern like such-and-such in the headers, this is probably a message from version xx.yy of Ken Pogran's Multics mailer." With enough such tests, it formed an opinion about which mail daemon it was talking with, and which mail UI program had created a message.
>
> Having hopefully figured out the other guy's genealogy (and therefore protocol dialect), the code then acted based on a painfully collected set of observations about how that system behaved.[52]

RFC-680 is notable for documenting the increase in header fields that had taken place over two years. It defined a number of widely used but not standardized header fields, including most notably, the TO field, but also CC (carbon copy), BCC (blind carbon copy), IN-REPLY-TO, SENDER, and MESSAGE-ID. Introduction of the TO field meant a format needed to be chosen for sending to multiple recipients. The proposal called for multiple email addresses in a field separated by commas. The RFC also documented the use of @ instead of "at."

RFC-680 was a clear step forward from RFC-561. Still, RFC-680 had limitations. It was based on practices on TENEX systems, which were not always representative of the Arpanet community as a whole. (For example, the decision to separate addresses in the TO field with commas was a TENEX convention.) Its syntax had bugs (it unintentionally permitted "@" and comma in mailbox names). Furthermore, pragmatically, RFC-680, while intended to become a standard, was never officially issued as a standard.[53]

In addition, RFC-680 revealed a philosophical split between members of the Message Services Committee. The MIT members (Vezza

and Haverty) felt email headers were primarily of use to the email handling programs and should be designed to be machine-readable. Others felt that headers should focus on being human readable. RFC-680 tried to strike a compromise, which apparently pleased neither side.[54]

The result was confusion. Some sites updated their mailers to conform to RFC-680 while others continued to follow RFC-561.

A new standard

Sometime in 1976, the Message Services Committee was replaced by the ARPA Committee on Human-Aided Communication.[55] One of the new committee's early actions was to seek to clarify the state of standards for email message formats. A vigorous email discussion on the Header-People mailing list in the fall of 1976 led to a new proposed standard in RFC-724 (''Proposed Standard for Message Format'') written by Ken Pogran (MIT), John Vittal (now at BBN), Dave Crocker, and Austin Henderson.[56] It came out in early 1977.

The RFC-724 authors, like the RFC-680 authors, sought mostly to document current practice. Vittal nicely summarized the goals as:

> to take RFC680 plus what we felt were things which people were already doing that were useful to most, take out some things that weren't terribly useful and probably shouldn't have been in 680 in the first place, and come up with a new specification. There were several things that some systems were already doing: comments (e.g. the day of week in parentheses), association of people names with user names (like at places like Stanford, CMU and MIT, also using parenthesization), random date format preferences (Multics vs Tenex, etc.), and so on. Elements of 680 which were not perceived as necessary were mostly the military-like field names such as precedence, as well as syntactic inconsistencies (bugs), and syntactic limitations. These could all be accomplished by using the notion of user-defined fields.[57]

RFC-724 defined a text-only message format. The message header and contents were ASCII. The authors observed that, at some point in the future, clearly email would use richer binary formats, but that was beyond the immediate need.

The new RFC provoked a tremendous amount of debate on Header-People and a more focused (and very distinct) discussion on MsgGroup.

The MsgGroup discussion raised two issues. First, was the new RFC going to cause much longer message headers that users would have to see? Second, wasn't the major issue simply a desire to embed users' real names into TO and FROM fields and, in that light, were all the other header fields necessary? The conclusion was that extra header information simply reflected the reality of what had already happened, and the desire not to see them pointed to a need for user agents to edit header information, and that yes, adding names mattered.

The Header-People debate was rooted in specification details. The best example of the tenor of discussion is a multiday argument (rich with ad hominem remarks) about whether to use 12-hour or 24-hour times in the DATE field, with much debate about whether ''12am'', ''12pm'', or ''12m'' was the correct abbreviation for midnight. The upshot was to eliminate support for 12-hour times.[58]

The result was RFC-733, a revision (by the same authors) of RFC-724. The major improvement in the revision (beyond the date field) was a clear statement of how to include names with email addresses. The format was to put the email address in angle brackets (< >) as in ''David H. Crocker'' <crocker@rand-unix>, and if the text before the brackets contained any special characters such as punctuation or control characters, it had to be in quotes. The RFC also made clear that mailing lists looked like any other mailbox.[59] Issued in November 1977, RFC-733 was the official standard for message formats for five years, and a de facto standard well into the mid-1980s.

Today's standard

In 1982, as the email community was preparing to transition to the Internet, the authors of RFC-733 were asked to update it. The authors of 733 had several conversations about what the changes should be, but only Dave Crocker (who had become a graduate student at the University of Delaware) had the time to undertake the revisions. Several features of RFC-733 that had failed to win popular acceptance were deleted, and three new fields, FORWARDED, RESENT-FROM, and RESENT-TO, were added (to support the common practice of forwarding an email message to someone else).

A more startling feature (in retrospect) was the addition of the RECEIVED field. RECEIVED is odd because it, alone of all the fields in the message header, was created by MTAs rather than UAs. Every MTA was required to insert a RECEIVED field into the message, to track the message's path through the network. Looking

back, this is an odd and subtle architectural change that made MTAs responsible for understanding the format of messages, which previously (ignoring the practical problem of address rewriting; see the next section) MTAs had not needed to understand.

The result, written by Crocker and published in August 1982, was RFC-822. RFC-822, or more commonly, simply 822 format, remains the basic standard a quarter century later. (An updated version appeared as RFC-2822 in 2001, but the basic format is unchanged.)[60]

Before we leave the discussion of the evolution of message formats, a few observations are in order. First, developing a message format was a difficult intellectual problem. RFC-822 is 47 pages long and a combination of an augmented Backus-Naur notation that defined each field's format and briefly stated each field's semantics. It is comparable in complexity to the computer language specifications of the time. Second, it is hard to understate the importance of RFC-733. RFC-733 came out early enough to become the de facto standard for email message formats throughout much of the world. The UUCP network, the Computer Science Network (CSnet) and Bitnet all ended up using RFC-733 format for their email messages.[61]

Evolving the MTA

SNDMSG was the earliest MTA. It simply delivered the message or returned an immediate error message saying it had failed. After about a year, Bob Clements enhanced SNDMSG to retransmit messages if the remote host was down.[62] About two years later, SNDMSG was updated to place each message in a file in the user's directory (one file per email) and a new program, called MAILER, would periodically pick up and deliver email files in the user's directory.[63] (Observe that this change converted SNDMSG to a user agent, with MAILER taking on the role of MTA.)

In a nutshell, that incremental evolution describes the experience of developing MTAs in the 1970s. Each operating system would implement an MTA, which was then refined over the years to deal with environmental conditions.

Unfortunately, the different MTAs evolved differently. The underlying problem was that email via FTP was underspecified. (It is useful to observe that the specification for email delivery with FTP was two pages long, while the SMTP specification, when it appeared, was 68 pages long.) Implementers had considerable latitude, and they used it.[64] By the mid-1970s, implementing an MTA was getting harder, not because email had become more difficult, but because the profusion of slightly different MTAs meant that everyone's MTA had to be programmed to deal with the differences.

For example, there was considerable disagreement about whether one had to login to the remote system (FTP had a login command called User) before trying to deliver email with MLFL. Multics required a login. TENEX did not. So MTAs had to include code to recognize when they were talking to Multics and when to TENEX and adapt their behavior accordingly.

SMTP, because it was well-specified, eventually solved this problem (see the ''SMTP and avoiding second system syndrome'' section). Unfortunately, by this point, a new problem had arisen: multiple email networks.

Bitnet, CSnet, and UUCP

Between 1978 and 1981, three major email networks were created. Although the Internet remained the largest network throughout the 1980s, these three networks (UUCP, CSnet, and Bitnet) would grow big enough to influence email standards. The UUCP network was comparable to the Internet in size. And, almost from the start, the four networks were interconnected,[65] creating massive challenges for MTAs of routing between four networks (not counting the smaller networks that appeared) with different address formats.

UUCP network. The UUCP network (named for the Unix-to-Unix CoPy program over which it was built) began inside AT&T in 1978.[66] It used dial-up telephone links to exchange files and within a few months was moving email. AT&T soon distributed the software and the UUCP network, made up of cooperating sites, was off and running. Over the next decade it grew at a prodigious rate, such that by 1990, its population was estimated at a million users—comparable to the Internet's population.[67]

The UUCP network was a multihop network. To reach machine V, an email from machine M might have to pass through intermediate systems Q and T. The motivation for this approach was to minimize phone bills. In the 1970s and early 1980s, long distance calls were expensive, and the rates differed by hour (with evening and night rates being sharply lower). Modems were slow (a couple hundred bytes per second was considered good) and files were (relatively speaking) large.

So the typical operating mode at any UUCP site was to save up all email until 5 p.m., then call a nearby UUCP site to forward email along and receive inbound email. Indeed, over the course of the night, several phone calls would be made to push outbound mail and receive inbound mail. Depending on the calling schedules and the connectivity of the machines, email could travel a few or several hops before the nightly calling frenzy ended.

Initially, the person composing the email had to spell out the entire path a piece of email needed to take through the network. In the UUCP network, the hops were separated by exclamation points (''!,'' pronounced as ''bang''). So, someone mailing the author via UUCP from UC Berkeley in the 1980s would send it to *ucbvax!ihnp4!harvard!bbn!craig* (in which each text string followed by a ''!'' is known as a hop; this example has four hops).

In 1982, Steve Bellovin wrote *pathalias,* a tool designed to compute paths from a network map. He refined it with Peter Honeyman.[68] Pathalias was distributed widely. Now, by keeping a map of regional connectivity, it became possible to email via landmark sites and have them fill in the missing hops. So, for instance, the author's address could be reduced to *ihnp4!bbn!craig* and the *harvard* hop would be dynamically inserted.

In 1984, Mark Horton began an effort to create a complete UUCP network map, which reached fruition about 1986. After that, UUCP users could simply type *sitename!user,* and *pathalias* would compute a path to *sitename* for them. An even fancier trick was to add a network domain to the sitename, such as *bbn.arpa!craig*, and *pathalias* would compute a path to an email gateway between the UUCP network and the Internet.

CSnet. By the late 1970s, the computer science research community realized that the Arpanet was changing how people did research. Researchers who had access to a network got information more quickly, and could collaborate and share work more easily. Thus was identified the first ''digital divide''—between computer science departments that had access to Arpanet and those that did not.[69]

The goal of the Computer Science Network (CSnet) was to bridge that gap. Created in 1981 by the National Science Foundation in cooperation with ARPA, CSnet linked computer science departments and industrial research laboratories to the Arpanet (and then the Internet).[70]

CSnet was designed to become self-supporting. The ARPA and NSF funding was only to provide start-up capital and an initial operations budget.

CSnet was designed to become self-supporting. The ARPA and NSF funding was only to provide start-up capital and an initial operations budget. For the first two years, CSnet operations were distributed between the University of Wisconsin and the University of Delaware, with help from RAND (which ran a gateway on the West Coast). Beginning in 1983, the network was operated by BBN, where a team of roughly 10 people provided technical support (including writing or maintaining much of the email software used by CSnet members), user services, and did marketing and sales. By 1988, CSnet was self-supporting and had approximately 180 members, most of them computer science departments in North America.

Technologically, CSnet did everything possible to make its members feel part of the Internet community. Initially, connectivity was almost entirely email only, using dial-up phone service. Over time, direct access via IP was also supported over a variety of media, including IP over X.25[71] and the first dial-up IP network.[72]

After 1983, email in CSnet all went through a single email gateway, CSNET-RELAY, which sat on both CSnet and the Internet. Email was routed by addressing it to the relay, with the user address being the target address on the other network. The syntax used a percent sign (%) to divide the next hop user name from relay address. So, to get from the Internet to a CSnet host, one emailed to *user%host.csnet@csnet-relay.arpa*. From CSnet, one emailed *user%host.arpa@csnet-relay.csnet*. Email was formatted according to RFC-733 and 822 standards.

Bitnet. Bitnet was established in the same year as CSnet, but with a different driving force. Bitnet (''Because It's There'' or, later, ''Because It's Time'') was created by

university computer centers (now information technology offices) to interconnect their computing facilities with email and file transfer. Because the centers typically used IBM mainframes running the VM operating system, Bitnet was constructed from low-speed leased lines running IBM networking software, on which email was overlaid.

Like CSnet, Bitnet used Internet email standards (with the %-hack in the email address for gatewaying). Unlike CSnet, Bitnet did not have a central management or support center. Instead, most functions were volunteer activities, with coordination provided by Educom (Interuniversity Communications Council). In mid-1988, Bitnet had nearly 400 member sites.

The boards of Bitnet and CSnet overlapped and the two networks eventually merged, so one may wonder why they were distinct in the first place. The distinction lies in the relationship, often contentious, between computer science departments and computing centers in the 1970s and 1980s. Computer science departments typically maintained their own computing facilities, to enable research by computer science faculty. Computing centers were university-wide resources that sought to provide stable computing environments for researchers in other disciplines. The stereotype was that computer science departments ran cutting-edge operating systems on minicomputers and workstations while computing centers ran established commercial operating systems on mainframes. More important, from an institutional perspective, the computer science department typically provided a haven for those on campus who were (for whatever reason) disgruntled with the computing center. Neither party particularly wanted to rely on the other for network access, with the result that there were two networks: one for each community.

Email addressing across networks. The four networks (including the Internet) periodically viewed themselves as competitors. Yet the four networks were also committed to making email work among them. A number of sites brought up gateways between the networks. Even more sites made a point of residing on more than one network, to ensure ease of mailing for their users.

It is widely agreed that, by the early 1980s, email addresses were a disaster both for users trying to email across networks, and network administrators trying to keep the email flowing.

The disaster had two dimensions. First, one had to know which network a user was on. For instance, if someone told you he was *bob@princeton,* one had to immediately ask "which network" because *princeton.bitnet* and *princeton.csnet* were different machines and were not interconnected. If a user forgot, or her email software removed the network appellation (e.g., *.csnet*) the email would be delivered to the *bob@princeton* in whichever network the sender was in.

The second problem was that, even if one knew which network an email address was in, getting it there was not easy. To take a relatively common example, consider the following four addresses:

ihnp4!ucbvax!bob%princeton.csnet@csnet-relay.arpa
bob%princeton.csnet%csnet-relay.arpa@wiscvm
bob%princeton.csnet@csnet-relay.arpa
bob@princeton

These represent the four likely addresses for reaching *bob* at Princeton's CSnet host, from the UUCP network, Bitnet, the Internet, and CSnet respectively. If the examples are not painful enough, consider the first address and how it would be handled in transit.

It starts in the UUCP network and is passed to *ihnp4* (a key UUCP relay at Bell Labs in Naperville, Illinois). *Ihnp4* must puzzle out *ucbvax!bob%princeton.csnet@csnet-relay.arpa* and decide if the email address is to the left of the @ sign (Internet style) or to the right of the bang (UUCP style). As *ihnp4* is a UUCP-only system, it knows to use UUCP addressing and passes the message to *ucbvax* at the University of California at Berkeley. *Ucbvax* is a gateway on both the Internet and UUCP networks so it must puzzle out *bob%princeton.csnet@csnet-relay.arpa*. Thankfully, *ucbvax* was not on CSnet and clearly not the same system as *csnet-relay.arpa,* so *bob%princeton.csnet* is no good. Thus the message must be sent to the CSnet relay (and, because Arpanet did not strip mailing information, it remains *bob%princeton.csnet@csnet-relay.arpa*). CSnet's relay in turn extracts the address to the left of the @ sign, to get *bob%princeton.csnet* and delivers the email to Princeton.

Observe that there's ample chance for confusion. Another nasty problem was that each mailer had to make sure that the FROM address in the email was updated (and sometimes the TO and CC addresses as well) so that the recipient of the email could successfully reply to it. Yet another challenge was that, for a period, the United Kingdom decided to

reverse the order of labels in a domain name (so *Kirstein@uk.ac.ucl.cs*) with the result that some mailers had to parse names backward and forward (''bothways'' mode) to see if they made sense.

It is no surprise that the people who made major contributions to email MTAs at this time were people closely affiliated with email gateways.

delivermail, sendmail, and mmdf

The appearance of new email networks transformed the complexity of the MTA. Now, at least on systems that were on multiple email networks, the MTA had to understand multiple addressing formats and routing rules and competently move messages between the various networks as appropriate. One sign that the problem of writing an MTA had gotten hard was that it became the subject of serious academic research. The major contributions were made by two graduate students: Eric Allman at UC Berkeley (*delivermail* and *sendmail)* and Dave Crocker (who had left RAND to study at the University of Delaware, where he wrote *mmdf)*.

Both men were trying to solve essentially the same problem: supporting multiple email networks in one system. Allman needed an MTA for UC Berkeley's main email system, which served as the university's email gateway between the UUCP network and the Arpanet and local email delivery. Crocker needed an MTA to support local email, Arpanet email, and a new phone-based delivery system which eventually became CSnet's PhoneNet protocol. The two men solved the problem very differently.

delivermail. Allman's *delivermail,* the simplest of these MTAs, was written for Berkeley's BSD Unix operating system in 1979 and was a basic program[73] not greatly more complex in its workings than Bob Clements' 1973-vintage SNDMSG. When invoked by a user agent (or the inbound FTP server), *delivermail* expected to be given a message, which it would either deliver or return an error message. The big difference was that *delivermail* implemented a layer of indirection. Rather than delivering the message to a mailbox or a remote system, *delivermail* looked at the destination address and then picked a program to deliver the message to. So, for instance, to deliver Arpanet mail via FTP, *delivermail* called an auxiliary program called *arpa* and passed the mail to the *arpa* program and waited for a (real-time) response regarding delivery. If, by some mischance, the message had to be queued, *arpa* (not *delivermail)* would queue it.

To parse the address, *delivermail* used the simple expedient of assuming that an at-sign meant Arpanet mail, an exclamation point in the address meant UUCP, and a colon meant the local BERKNET protocols. For each address type, *delivermail* could be configured either to call a program to deliver the mail, or call a program to relay the mail to the appropriate gateway (one email gateway per type).

The *delivermail* MTA had a powerful aliases features, in which a destination address could be expanded to a list of email addresses. It also had a first class logging system (a way to record what *delivermail* did) called *syslog*. Email systems were developing increasingly sophisticated logging mechanisms; *syslog* was so good that it eventually became a standard part of BSD Unix and is now used by a wide range of applications.

One surprising feature of *delivermail* was that part of its configuration was compiled into the program. That is, for each machine, one compiled a custom version of *delivermail*. So, for instance, if the machine was connected to Arpanet, one compiled *delivermail* with the –DHAS_ARPA flag to the C compiler.

mmdf. About the same time that Allman was creating *delivermail,* Dave Crocker was writing the first version of *mmdf* (the Multichannel Memo Distribution Facility).[74] Rather than seek to process each message immediately, as *delivermail* did, Crocker sought to decompose the process into multiple stages.

When a message arrived (via the network or from a user agent), the message was given to a program called *submit,* which checked that the message format was correct (here the common use of 733 format was a big win) and then looked at the address to decide what network the message was to go out on. The message was assigned to a ''channel.'' Each channel had its own queue: a directory where messages and their ''envelopes'' (control information) were stored. Simply, *submit* placed the message in the right queue.

Another program, called *deliver,* was regularly scanning the queues for messages. When a new message appeared, *deliver* called on a channel-specific program (e.g., *mmdf*'s equivalent of *delivermail*'s *arpa* program for Arpanet email) to deliver the message. If message delivery failed, *submit* was called to send the message back to its sender. If there was a transient error (e.g., the remote host was

down), the message was left in the queue and *deliver* would try it again later.

The *mmdf* MTA also supported aliases and had a fine logging system.

An important contribution of *mmdf* was achieving an effective split of the message delivery process. Diagnosing email problems (whether configuration problems or problems with particular messages) was cleanly compartmentalized. Similarly, *submit* prevented junk from entering the system; *deliver* handled problems in delivery. An operator knew where the problem was by seeing which program was complaining in the logs.

Another contribution was restriction of privileges. One of the key problems in any mail system is that whatever program delivers mail to the user's mailbox needs special privileges. In *mmdf,* that was one small program, the local channel delivery process. All the other processes could be run as a regular user (usually called ''mmdf'').

The channel model also proved flexible. A message could go through multiple channels before leaving a system. Soon, *mmdf* developed a ''list'' channel to handle mailing lists. A message was placed in the list channel to have its destination address expanded. It exited the list channel by being placed in one or more channels to be delivered to members of the mailing list. Later, when MX resource records were introduced (see the ''Email routing with domain names'' section), they introduced a new error: a domain name that (because of DNS problems) could not currently be looked up. In *mmdf* this was trivially handled by creating a new channel, where *submit* placed messages whose addresses could not be resolved at the moment.

A downside of *mmdf* was that rather than one configuration file, there were several, scattered in different places. While each configuration file was simple (a list of attribute: value pairs), the sheer number of them could prove frustrating.

sendmail. Based on experience with *delivermail,* Eric Allman decided to write a new MTA for release with the 4.2 version of BSD Unix. The new MTA was called *sendmail.*

Culturally, *sendmail* was similar to *delivermail.* But from a practical perspective, it was quite different. Major differences included the following:[75]

- Configuration was determined by a file, called *sendmail.cf,* rather than being compiled in.
- The address parsing rules and message delivery rules were defined by a grammar in the configuration file.
- *sendmail* now maintained its own message queue.
- Certain delivery programs (most notably email delivery via SMTP) were compiled into *sendmail* instead of client programs (e.g., *arpa).*

But this list understates the transformation from *delivermail* to *sendmail: sendmail* was almost an order of magnitude more complex (measured in lines of code) and tremendously more flexible.

The changes had an interesting mix of consequences. Probably the most important consequence was flexibility. Placing address parsing and configuration rules in a grammar made it possible to dynamically configure *sendmail* for arbitrarily complex email environments.

Another consequence was a reinforcement of *delivermail*'s approach of putting all the email expertise into one program. SMTP was now embedded in *sendmail.* So too was queue management. It made *sendmail* a complex program and hard to change. Allman later noted that *sendmail* should have been better decomposed into constituent functions, even if only internally.[76]

An unexpected consequence was that crafting and debugging *sendmail*'s single configuration file (*sendmail.cf*) became a central preoccupation (some would say headache) for system administrators over the next several years. A properly working email system required the configuration file be right. And *sendmail*'s grammar (with a fondness for single-letter tokens, which made mnemonic naming impossible) gave administrators many opportunities to make a mistake.

Evolution and perspective

Over the 1980s, both *sendmail* and *mmdf* prospered: *mmdf* was substantially reworked by Crocker, Doug Kingston (of the Army's Ballistic Research Laboratory), Steve Kille (of University College London), and Dan Long and me (of BBN) into a new release called *mmdf2,* which was used at a number of major email centers in the mid- and late 1980s.

Also, *mmdf* inspired PMDF, a rewrite of *mmdf* in Pascal for the VMS operating system. The initial implementation was done by Ira Winston at the University of Pennsylvania. It was then maintained and substantially revised by Mark Vassol and Ned Freed (then at Oklahoma State University). PMDF became a

popular email system for Digital Equipment Corporation's VMS operating system and, over time, became the dominant email system on Bitnet (where VMS systems were popular) and eventually became a commercial product.

The sendmail MTA was repeatedly improved in subtle ways. Over time, as BSD Unix became the most popular operating system on the Internet, *sendmail* became the most common MTA, while the *sendmail.cf* file continued to build a reputation for being complex.

In the end, *sendmail* had the last word. When the complex mix of email networks consolidated into a single email network after 1990, it was no longer necessary to write a *sendmail* grammar that could handle multiple email address formats. The *sendmail.cf* became much simpler. *Sendmail* could be recognized for its flexibility without being forced to demonstrate its potential complexity. It remains a popular MTA to this day.

SMTP and avoiding second system syndrome

By 1980, the Internet protocols were rapidly maturing, and ARPA (rechristened DARPA some years earlier) had started to plan the operational transition from Arpanet to Internet protocols. Initially, the expectation was that Internet email would be multimedia and thus a full-scale replacement of Arpanet email with Internet multimedia email would be made.[77] However, by May 1980, Vint Cerf of DARPA had concluded multimedia email would come too late for the Internet transition and that the problem of supporting text mail and bridging email from systems using the old Arpanet protocols to systems using the Internet protocols needed a solution.[78] In September 1980, three RFCs appeared that were intended to start the process of planning the transition.

The first RFC (RFC-771) was written by Cerf and Jon Postel (at ISI) and was a plan to make the transition from Arpanet email protocols to Internet email protocols using designated email gateways that operated using both protocol suites.[79]

The other two RFCs are more interesting. RFC-773 was an addendum to RFC-771, written by Cerf, and sketched out some of the key technical issues in the transition. Cerf was concerned to make the email transition as simple as possible and to defer hard work until multimedia email was in place on the Internet. One surprising statement followed the observation that FTP-based transfer passed only the user part of *user@host* to the remote system, but email gateways needed to know the *host* part to effectively gateway email. Rather than bite the bullet and accept an Arpanet change to FTP to pass the *host* part, Cerf suggested that, for compatibility sake, the *user* part be standardized across the Arpanet/Internet—in effect, every system was to know every user's email name and where to deliver its mail! This idea seems to have been rapidly abandoned.

In RFC-772, Suzanne Sluizer and Jon Postel proposed a new "Mail Transfer Protocol" or MTP.[80] Its purpose was to serve as the bridge protocol for the email gateways between Arpanet and Internet email protocols.[81] What, precisely, the Internet email protocols would be was not discussed (though clearly the intent was they would be new protocols capable of supporting multimedia). Very little thought had been given to email protocols since the 1973 FTP meeting but MTP tried to take advantage of what little thought had taken place. In particular, MTP sought to provide better support for delivering email messages to multiple users on a single system.

In community lore, MTP is remembered as an ugly protocol. In truth, it was not ugly, but it was complex. It negotiated two different ways to send email. One could either send the message first and then send the destination address, or one could send the destination address and then the message. MTP used complex commands, which made understanding error codes difficult. For instance, it was possible to say MAIL <*from@host1*> TO <*remote@host2*> as a single command, which made it hard to parse error messages about addresses to determine whether the FROM or TO address (or both) was in error. It had an approach to email forwarding that permitted an MTA to announce that it didn't know how to forward a message but would hold onto the message anyway until the MTA's operator figured out where the message should go—a bizarre idea that would have ensured endless work for operators.

Some of these problems were noted at the time.[82] Nevertheless, MTP soldiered on. At least four implementations were made,[83] and the MTP specification was revised in May 1981. The revisions appear to be largely cosmetic and the protocol remained complex. The impression is that the email transition plans were poorly thought through. Some of the Internet researchers of the time remember that the community viewed email as a distraction—with so many problems in TCP and IP, who needed to look higher in the stack? They give credit to Cerf for forcing them to

periodically pay attention. Then, late in 1981, things suddenly cleared up.

The continuing criticism caused Postel to rethink MTP, and, in November 1981, he wrote an RFC describing a simpler protocol, the ''Simple Mail Transfer Protocol'' (SMTP). SMTP was, indeed, simple. Every command had zero or one arguments. Recipients were always listed before the message was sent, and each recipient was listed and acknowledged separately. A slightly revised version of the specification came out as RFC-821.

It is not clear what inspired SMTP's design, but there is a hint. Sluizer and Postel published two RFCs documenting their experience implementing MTP.[84] The first one, RFC-784, observed that it was convenient to maintain two files for each email message: a control file called the envelope and the message itself. At this point, the concept of an envelope was still relatively new. The term *envelope* had been coined in 1975 as a way of discussing header fields that MTAs needed to be able to deliver a message,[85] but by 1979, its meaning had shifted to mean the metadata associated with a message.[86]

The second RFC, RFC-785, detailed the internal structure of the envelope file as it appeared on TOPS-20. Each item of information (each email recipient, the email sender, etc.) was kept on a separate line. And if one reads the SMTP specification, each command in SMTP corresponds to adding a line of information in the TOPS-20 envelope file. So perhaps that is where Postel got his ideas for SMTP.

SMTP, with modest changes, remains the way email is transferred today, 25 years later. In that light, it makes sense to try to assess what has made SMTP so long-lived.

First, we note that SMTP did (and in some cases, still does) have deficiencies. Despite Postel's interest in the support of multimedia mail, SMTP was defined to use 7-bit ASCII—a decision that had to be undone a decade later. SMTP's request/reply format causes SMTP connections to follow a pattern of many little data exchanges (command/reply, command/reply) and then a big transfer of the actual email message. This pattern turns out to be bad both for TCP and on network paths with long delays. This problem was eventually solved with SMTP pipelining.[87] RFC-821 had some SMTP commands to send messages directly to user terminals. These commands were never implemented widely. In addition, SMTP has a small race condition, such that email can be duplicated.[88]

These deficiencies are outweighed by SMTP's design. Each command has zero or one arguments. Reply codes are three-digit numbers, with the first digit standardized.[89] A positive response or an error can be determined by the first digit, even if the particular error code is novel. Transmission of a message typically requires just three commands: MAIL FROM, RCPT TO, and DATA. While SMTP clearly reflects its roots in FTP (which has a similar command style), the complicating features of FTP (in particular, features for interactive user support) were removed.

Domains and a new way to route

From the early days of the Arpanet, the DDN Network Information Center (NIC) maintained a file, named HOSTS.TXT, which mapped single-word hostnames to network addresses. This file was copied to every host on the Arpanet and later Internet, so users could use character-based mnemonic hostnames such as *bbn-loki* rather than numeric addresses such as 128.89.1.178.

Unfortunately, the HOSTS.TXT scheme had several limitations. Updates were difficult. The NIC needed to be notified when a new host was installed or host information was changed, and then every host needed to download the new HOSTS.TXT. To minimize the network load (HOSTS.TXT was a relatively large file and having every host get a copy could cause considerable network load), HOSTS.TXT was updated just a few times a week. Simply getting new information propagated could take several days.

Furthermore, the namespace was flat. Only one machine could be named FRODO. Furthermore, names were relatively short (24 characters maximum[90]), so users had to become increasingly creative about hostnames as the network grew.

In 1982, the Internet community set out to replace HOSTS.TXT with a distributed database. Zaw-Sing Su and Jon Postel of ISI wrote a proposal for a new naming structure. The scheme created hierarchical names, with different portions of the hierarchy, called domains, delimited by a dot (''.'') in the name.[91] For example, under this scheme, F.ISI, would be the name of host F in the ISI domain. The naming scheme was clearly inspired by the recently developed Grapevine distributed naming system developed at Xerox PARC.[92]

In November 1983, Paul Mockapetris of ISI issued two RFCs[93] specifying a distributed database system to support a domain name system (DNS). Reflecting considerable com-

mentary on the NameDroppers mailing list, the proposed DNS supported multiple levels of hierarchy (vs. the two levels used by Grapevine and suggested by Su and Postel). The DNS stored information as resource records, where each record mapped a name to a typed value such as an IP address. A single name could have multiple records (so a host with multiple IP addresses would have one resource record for each IP address). Work began at ISI and UC Berkeley to implement the proposed DNS in TOPS-20 and Berkeley Unix.[94]

By the summer of 1985, both servers were working (at least experimentally). But several deployment issues remained unresolved, at least two of which were vital: how email was to work with the DNS, and how the namespace should be organized.

Email routing with domain names

In the summer of 1985, I joined the staff of CSnet and was asked to see what modifications needed to be made to CSnet's email software to support the (now working) domain name system. Mockapetris's DNS specification had created two resource records to support email routing, but the specification only loosely specified how they would be used. Initially, I thought the problem would be easy[95] only to realize a few weeks later that there was a serious issue.[96]

Mockapetris had defined two email routing records for the DNS: a Mail Destination (MD) record and a Mail Forwarder (MF) record. The notion was to allow a domain name to specify that all email addressed to the domain was to be delivered to a particular host (an MD), or that the email could be relayed via one or more email gateways (MFs). The central idea here was new and powerful: under the DNS, the right side of the @ sign in an email address was no longer the host to which email was to be delivered, but a name for which email routing was specified.

I eventually realized that, if a name had both MD and MF records, there were situations where email could loop or worse, fail to be delivered.[97] I wrote a draft RFC describing a complex set of rules that ensured such failures would not occur and sent it to Jon Postel and Mockapetris. Postel and Mockapetris felt the proposed rules were ugly and burdensome to MTAs. They asked me to work with a small group of people, including Mockapetris, to find a better solution.[98]

After a couple days of discussion, Mockapetris suggested a potential solution was to have one record for mail routing, called a Mail EXchanger (MX) record. I worked through the details of the idea and crafted the routing rules for MX resource records. I reported that the resulting specification was indeed much simpler (about half as long as the previous one). Postel declared a solution had been found, and asked Mockapetris to update the DNS specification. Mockapetris' update and my specification appeared in January 1986.[99]

Another open question in late 1985 was what the top-level domains would be. As the DNS began to work, finalizing top-level domain names became an increasingly vital.

MX resource records remain the way email is routed today. The basic idea is simple. A name is associated with one or more MX resource records. Each resource record has the name of one host and a preference number. To route to a name, a mailer looks up the name's MX records and then successively tries to deliver to the hosts, starting with the host with the lowest (best) preference number, until delivery succeeds or the mailer runs out of MX records. To prevent loops, if the mailer is one of the MX hosts listed, it may only deliver to MXs with a lower preference value. Despite the simplicity, the scheme supports most useful types of email routing easily.[100]

Defining the domain name space

Another open question in late 1985 was what the top-level domains would be. Top-level domains are the last part of a domain name: thus in *example.com* the top-level is *.com*. As the DNS began to work, and as email was being modified to use it, the issue of finalizing top-level domain names became an increasingly vital issue.

It soon became clear that the issue transcended the Internet community. The major email networks connected to the Internet saw a chance to make their naming schemes

consistent. Indeed, one of the people pushing most vigorously for resolution was Dick Edmiston, who led CSnet.

Elizabeth ''Jake'' Feinler, head of the DDN NIC, hosted a two-day meeting at SRI International in late January 1986 to resolve all outstanding issues. Beyond several Internet representatives, mostly notably Postel, Mockapetris, and Ken Harrenstein (SRI), the meeting included representation from the UUCP community (Mark Horton), Bitnet (Dan Oberst), and CSnet (Laura Breeden and me), representatives (Kevin Dunlap and Jim Bloom) from the UC Berkeley BSD Unix project (which maintained *sendmail* and the *bind* DNS software) and Steve Kille (of University College London).[101]

Once the meeting began, it was clear that but for an odd issue about creating *.net*,[102] the real issue at the meeting was email compatibility. CSnet used Internet email standards whenever possible and planned to implement DNS naming throughout its network. The UUCP network, limited by its flat namespace, also saw an advantage in adopting domain names. Bitnet was less certain, but still felt domain names were of interest. More generally, a brief discussion of the routing technologies of the different networks made clear that it was possible to create seamless support for email addresses of the form *user@domain-name* that spanned the four networks. The end of the era of *ihnp4!ucbvax!bob%princeton.csnet@csnet-relay.arpa* was visible and exciting. Everyone at the meeting agreed to push to get their respective software ready.

Except … except that Mark Horton wanted compatibility with X.400 email addresses too. (X.400 is described in more detail in the ''X.400'' section). The X.400 naming system was known (though not yet working). It used names that were close to domain names. Steve Kille and Horton had worked out a way to map between X.400 addresses to DNS names, if the DNS followed certain naming practices. Horton wanted to make it possible for sites to pick domain names that would be compatible with X.400. Those questions led to the question of whether there would be a *.us* domain. X.400 names were assigned by country, and thus organizations in the United States, in the X.400 system, would have names ending in *.us*. If Kille and Horton were to achieve the goal of compatibility with X.400, there needed to be a *.us* domain, and names in the *.us* domain had to be given out according to X.400's rules.

Postel was adamantly opposed to structuring *.us* to fit X.400. He felt that forcing people to add a country code to their email address was much like forcing them to add a network name such as *.arpa* or *.bitnet* to their email address. In his view, both practices were ugly and restrictive.[103] He asked why a university had to make the top level of its name the country in which the university was situated, when clearly the most important aspect of the institution was that it was an educational organization. Equally vigorously, Postel had no interest in assisting a conversion to X.400. Indeed, he already had taken the step (as the Internet Assigned Numbers Authority) of assigning control of *.us* to himself and made it clear that his naming structure for *.us* would bear no relationship to anything compatible with X.400.

The debate ran, on and off throughout the meeting. In the end, the parties agreed to disagree, but accept that the decision was Postel's.

At the time, Postel's intransigence seemed just a stubborn attempt to delay an inevitable transition to X.400. In retrospect, several factors were about to converge to make the debate, arguably, X.400's high water mark.

By standardizing on domain names, the meeting created a common email addressing and message format that probably covered over 90 percent of the email community of the time. Over the next several years, organizations on CSnet, Bitnet, and UUCP, already using domain names and thus culturally acclimated to the Internet world, would begin seamlessly transitioning to the Internet. SMTP, RFC-822, and domain names were about to become a technical juggernaut that X.400 would be hard-put to displace. Postel's decision to keep *.us* distinct from X.400 made the process of replacement by X.400 tougher—everyone would have to change email addresses (precisely the barrier that the meeting had eliminated for organizations on CSnet, Bitnet, and UUCP who wished to join the Internet).

If X.400 was to become the next email standard, it now had to pin its hopes on the fact that X.400 supported multimedia while SMTP/RFC-822 did not.

Long tough path to multimedia (e)mail

Multimedia mail is email that contains a richer set of objects than simply ASCII text. Throughout the 1970s, email on the Arpanet and most other email systems was limited to ASCII. At the end of the 1970s, researchers and implementers began to think about how email might be enriched.

Early multimedia

In 1977, IFIP created a working group (WG 6.5) to address the need for standards for computer-based message systems. Grossly simplifying its charter (which included dealing with issues such as networked Telex messages), the working group was to lay the groundwork for an international standard for email. This effort was eventually to lead to the CCITT/ISO X.400 standards for email. Real work seems to have begun sometime in 1978 or 1979. As part of this effort, Debbie Deutsch and John Vittal at BBN were thinking about the format of email messages.

Under the auspices of the National Software Works program,[104] Jon Postel at ISI started investigating protocols to move multimedia messages between systems. In March 1979, Postel published ideas for an "Internet Message Protocol."[105]

In 1978, the UUCP email network began operation. In 1979, responding to a need for a way to safely send binary files between systems, Mark Horton (then a grad student at Berkeley) wrote the *uuencode* program, and it was distributed with the 4.0BSD distribution of the Berkeley Unix operating system. *uuencode* converted binary files into a formatted ASCII file that could be included in any email message. A complementary program, *uudecode,* could read the formatted ASCII and extract the binary contents. *uudecode* was cleverly designed to skip over any leading text until it hit the line encoding the start of a *uuencode*d object. So you could include some leading text in the email describing the binary object being sent and yet safely feed the entire email to *uudecode* to extract the binary. *uuencode*'s major contribution to multimedia mail was to demonstrate that people did want to email around binaries—for years, on the Internet, *uuencode* was the way binary data was sent.[106]

Oddly, while the work on *uuencode,* Postel's work at ISI, and the work at BBN all led to fruitful results, it is hard to directly trace the work in any of them to the final development of Internet multimedia mail. But they created a milieu in which multimedia mail was anticipated, and finally, after long effort, achieved.

Internet multimedia—Round one

By 1980, Postel's work[107] on multimedia protocols had created an expectation that the Internet would shortly transition to multimedia email. As the introduction to the 1980 email transition plan makes clear:

> This plan covers only the transition from the current text computer mail in the Arpanet environment to text computer mail in an Internet environment. This plan does not address a second transition from text only mail to multimedia mail.[108]

Cerf's commentary on the transition plan noted:

> DARPA is beginning a new phase of research into automatic electronic message handling systems. Ultimately it is intended that electronic messages incorporate multiple media such as text, facsimile, compressed digitized voice, graphics and so on.[79]

By the start of 1982, there were at least nine Internet multimedia projects at seven institutions (CMU, ISI, MIT, COMSAT, BBN, UCL, and SRI). The jump in effort was sparked by the advent of desktop machines with high-quality graphics. Several projects were using the new PERQ workstations, MIT was using Apollo workstations, SRI was using the Foonly F-5 (a desktop PDP-10 clone), and BBN was using its own graphics workstation called the Jericho. Most of the research effort was devoted to trying to figure out how to encapsulate voice and CCITT fax data into email.[109]

Despite the large number of efforts and the apparent interest in getting multimedia email working over the Internet, little came of these projects. The most successful appears to have been the Diamond multimedia project led by Bob Thomas and Harry Forsdick at BBN. (Also on the team was Ray Tomlinson.) By 1985, Diamond had a complete multimedia email system with user interface, mail transport system, and a multimedia editor to create documents that blended voice, video, spreadsheets, and other data.[110] BBN made Diamond into a product (called Slate) and sold a modest number of systems.

Impressive as Diamond was (and the demos were wonderful), it proved a developmental dead-end for two reasons. The first was simply that Diamond (like the other multimedia projects) was too soon. As Cerf's comments about incorporating voice and fax into regular emails show, there was a shortage of digital data. The profusion of digital, often graphics-rich, data was still a few years away. The second problem was that when digital data became available, users turned out to want to pick their own tools. That is, rather than use Diamond's built-in spreadsheet editor on a Diamond document, they wanted to use Excel or Lotus 1-2-3 to create a

document and then email that document to their colleagues. In short, the challenges for multimedia email were not those of creating content, but rather those of packaging binary objects or ''attachments'' into regular email, and how to create open interfaces that allowed applications (and email user agents) to insert and extract those attachments from email easily.

A slightly later (1987) and more successful activity was the messaging system for the Andrew Project at CMU. The Andrew project was a collaboration between IBM and Carnegie Mellon University to create a powerful and affordable computing environment for students.[111] Andrew sought to create a seamless computing environment throughout campus, where students could log into any machine and read and write email, do coursework, or any one of a number of other activities. The Andrew Message System (AMS) was designed to be a showcase application demonstrating the utility of Andrew.

In many ways, AMS was very similar to the earlier Internet projects (of which its designers appear to have been unaware).[112] It had its own multimedia editor, a custom GUI, and was built atop a sophisticated distributed system. But AMS had one important cultural difference: it was designed to coexist with existing email services rather than replace them. As a result, AMS's designer, Nathaniel Borenstein, sought to find ways to make AMS's multimedia messages compatible with *sendmail* and SMTP. That mind-set was to prove useful a few years later.

X.400

Interestingly, the people working on the CCITT/ISO email standard, called X.400, better understood the multimedia challenges and set out to create an email standard designed to carry third-party documents. In many respects, X.400 was one of the best CCITT/ISO networking standards activities. This success may be attributed to several email-savvy people who worked on it including John Vittal and Debbie Deutsch (both at BBN) and Jim White (by then at Xerox).

The X.400 team sought to design a completely new email system, built on top of the emerging ISO standards for data networking. To that end, they created an email delivery architecture (defining user agents and message transfer agents), and developed protocols for delivering email from end point to end point, and formats for email addresses and email messages.

A good example of the work, and an illustration of its quality and some of the challenges of the time, is the work on encoding data in messages.[113] X.400 chose to standardize on a binary data format for messages. That decision created a number of challenges including:

- *How to represent binary data efficiently on the network*. At the time, network capacity was extremely expensive, so there was a motivation to save every possible byte (and bit). The X.400 team sought a compact data representation.
- *How did applications embed data in messages?* The issue here is that data formats on different computers may be incompatible. They were certainly incompatible in the early 1980s, when a byte could be 5, 6, 7, 8, 9, or 10 bits long depending on the system. Data to be sent over the network needs a standard, ''external'' format that is computer independent. In the early 1980s this concept was alien to most programmers.[114] The X.400 approach was to encourage programmers to define how to move their data into and out of a generic format called an external data format.
- *Data and not garbage*. X.400 envisioned a world in which user agents developed new features and new applications would embed data in a document and so the receiving user agent might receive an email message that contained header fields it did not understand and application data from an application it had never seen before. How to ensure the user agent didn't simply report it had received garbage? The solution that X.400 chose to this problem was to make the data self-describing.

The result was a conceptually elegant encoding. Every piece of data is encoded as a triplet of a type (for self-description), a length (which permitted compressing data into its shortest representation), and a value. The encoding is recursive, so a structured type is a triplet, whose value field contained triplets for the individual fields in the type.

This general, self-describing, external data format initially was issued as the CCITT X.409 standard but soon became the standard known as Abstract Syntax Notation 1 (ASN.1). The compact self-describing data format was designed by Debbie Deutsch, Bob Resnick, John Vittal, and Jan Walker, working under a contract to the National Bureau of Standards.[115] To the encoding, Jim White added a formal

language intended to make it easy to specify a data representation without having to actually write out the bit-by-bit descriptions.

While X.400 did not survive, X.409/ASN.1 is widely used in network protocols. The communal consensus is that the formal language is a nuisance and the focus on encoding efficiently made the formatting overly complex. But the self-describing triples are elegant and solve many problems.

The X.400 community was justly proud of their work. An obvious question is why did not X.400, partly completed in 1984 and updated in 1988, become the Internet multimedia email standard?

The short answer is that it could have.[116] Steve Kille, who had been on the UCL multimedia project, concluded X.400 was the way to go and invested considerable effort in trying to make X.400 Internet-ready. However, there were challenges. X.400 was tightly embedded in the ISO standards (which were intentionally different from the Internet standards), and fitting X.400 into the Internet's email system was hard.

In addition, as the Horton-Postel debate of the previous section shows, there were political issues. The ISO/CCITT community was acutely aware that in X.400 they had produced a cutting-edge data networking standard for the Internet's key application (email) and hoped to ride the success of X.400 to convince (force) the Internet community to adopt the rest of the ISO "Open Systems Interconnection" (OSI) protocol suite in place of TCP/IP. Conversely, the Internet community was willing (sometimes grudgingly) to admit that X.400 was a nice piece of work. But most members of the Internet community also tarred X.400 as a component of the unpopular OSI protocol suite.

Internet multimedia—Round two

As the 1990s began, the Andrew Messaging System was in use at CMU and a derivative was available from Next Computers as NeXTMail. X.400 had undergone a round of revisions in 1988. But the Internet still lacked any way to send multimedia email. Binary data were, however, being routinely sent using *uuencode*.

In its meeting of December 1990, the Internet Engineering Task Force (IETF) decided to investigate the possibility of making SMTP "8-bit friendly," that is, making it possible to move binary information via SMTP.[117] Much of the interest in this change came from Europe. The European portion of the Internet was growing rapidly, and Europeans very much wanted to be able to send email in their own languages and character sets. At the time, SMTP limited them to (essentially) US ASCII.

At its next meeting in March 1991, the IETF effort both made tremendous progress and stumbled.[118]

The progress was to realize that the job was to extend SMTP *and* to extend RFC-822's email message format to support national character sets and binary (multimedia) material.[119] A group to study RFC-822 extensions was created. It promptly coalesced around a proposal from a team led by Nat Borenstein and Ned Freed. Borenstein, now working at Bellcore, had both the experience and credibility of having built the Andrew Messaging System. Freed brought several years of experience maintaining PMDF.

The IETF's stumble came in extending SMTP. By March 1991, there was uncertainty about the goal of the upgrade. The motivation in December 1990 had been to meet European needs, but now the new SMTP group (distinct from the 822 group) seemed to think enabling a transition to X.400 was a more important goal. Further, the details of upgrading the existing SMTP infrastructure to support 8-bit transfers were difficult and fraught with transition challenges, which worried vendors.

Over the summer of 1991, the two groups' paths diverged.

The group working on 822 extensions arguably had the harder problem. It had decided to adopt a scheme where binary objects were encoded as separate sections of the body of an RFC-822 message. This solution required devising a scheme for identifying the separate sections (the core idea of the Borenstein/Freed proposal) and then coming up with a uniform naming scheme that made it possible to identify what each binary object was and how it was encoded. The group had to resolve problems such as naming schemes for 200+ character sets. Yet, the group made swift progress and by late 1991 was making largely minor changes to a suite of documents recognizably defining the Multipurpose Internet Mail Extensions (MIME) standard that is used today. (In one amusing moment, the group agreed it did not want to support *uuencode* coding as it was distasteful, even though *uuencode* was the default way to send binary documents at the time.[120])

In contrast, the group working on 8-bit friendly SMTP floundered. Every solution presented challenges, and the group was struggling to make a choice. Furthermore, a significant part of the group felt that it was

time to replace SMTP (the "new protocol" approach) or transition to X.400. The effort lacked focus.

At the November 1991 IETF meeting, senior members of the community stepped forward to force a solution. John Klensin, whose networking experience stretched back to early Internet days, was induced to step in as the group's chair.[121] Klensin had the seniority and credibility to issue an ultimatum: either the group converged immediately on an approach or the 8-bit SMTP effort would be terminated. The ultimatum effectively excluded X.400 and new protocols from the agenda, leaving the group to grapple with the challenges of extending SMTP.

There were two key issues. First, how to transition from 7-bit to 8-bit gracefully. There was much discussion about how 7-bit MTAs should interact with 8-bit MTAs and vice versa, including questions of whether 8-bit MTAs needed to be able to convert messages from 8-bit to 7-bit representations (a painful idea). In the end, the decision was that 7-bit MTAs would refuse 8-bit email, and the 8-bit MTA had the choice of converting from 8-bit to 7-bit MIME or returning the email as undeliverable. The choice to permit email to be returned assumed that the general transition to 8-bit SMTP wouldn't take very long (as, indeed, it didn't).

The second issue was how to mark email messages as being 8-bit. Initially the idea was that SMTP would acquire a new set of commands to support 8-bit email (distinct from the 7-bit commands). During the winter of 1992, the group discussed the meanings of commands named CPBL and EMAL to support delivery of 8-bit emails.[122] Sometime in the spring of 1992, encouraged by Marshall Rose to find a simpler solution, the group members realized that these commands were superfluous.[123] The existing SMTP commands could be made to work with 8-bit email and all that was needed was a message at the start of an SMTP interaction to confirm that both ends of the conversation were 8-bit capable. The EHLO (extended HELO) message was promptly invented and the problem of SMTP extensions was then, largely, solved. RFC-1426 written by Klensin, Freed, Rose, Einar Stefferud, and Dave Crocker appeared in February 1993.[124] With modest modifications it defines what is today's standard.

A subtext to the IETF process is how many senior email experts were pulled into the process. While the December 1990 decision to update SMTP was made by a group with limited email expertise,[125] subsequent meetings were typically filled with email experts such as Nathaniel Borenstein, Mark Crispin, Dave Crocker, Erik Fair, Ned Freed, Christian Huitema, John Klensin, and Einar Stefferud.[126]

Closing thoughts

One of the interesting things about the history of Arpanet/Internet email is how often little issues were redirected into bigger, more important results. Dick Watson wanted to print memos on remote printers. Instead, Ray Tomlinson created networked email. Vint Cerf and Jon Postel wanted to make sure email was gatewayed between Arpanet and Internet protocols, yet the result was replacing FTP with SMTP. A desire to support European character sets started a process that, finally, caused the Internet to support multimedia email and attachments. A subtext to this process is the willingness to discard partial solutions such as MTP, or MD and MF resource records, for a better solution.

Another observation is the exceptional talent that was often involved. Several people mentioned have received the IEEE Internet Award (Dave Crocker, Steve Crocker, Paul Mockapetris, and Ray Tomlinson), the IEEE Kobayashi Award (Vint Cerf and Van Jacobson), or the ACM SIGCOMM Award (Cerf, Jacobson, Mockapetris, Jon Postel, and Larry Roberts) for their contributions. Many more are IEEE or ACM fellows.

Acknowledgments

In the 1970s, many key ideas never made it into an RFC or even an email archive. Thus writing this article required help from several people (in the form of interviews or reviews) to fill in the blanks. Thanks are due to Eric Allman, Steve Bellovin, Bob Braden, Jerry Burchfiel, Noel Chiappa, Dave Crocker, Steve Crocker, John Day, Peter Denning, Jake Feinler, Ken Harrenstein, Mary Ann Horton, Steve Kille, John Klensin, Alex McKenzie, Mike Padlipsky, Suzanne Sluizer, Ray Tomlinson, Al Vezza, John Vittal, Steve Walker, Barry Wessler, and Martin Yonke. In some situations, recollections differ, and I have been unable to find contemporary documentation to sort out the differences. Where the recollections are matters of nuance, I sought to present a middle ground. Where the differences seemed likely to be material for a future historian, I have documented differences in the notes. Any errors are, of course, my fault. I am intensely grateful to the BBN Library staff

(Jennie Connolly and Penny Steele-Perkins) for their invaluable assistance finding older references.

References and notes

1. J.S. Quarterman, *The Matrix: Computer Networks and Conferencing Systems Worldwide*, Digital Press. 1990; P.H. Salus, *Casting the Net: From ARPANET to Internet and Beyond*, Addison-Wesley, 1995; K. Hafner and M. Lyon, *Where Wizards Stay up Late*, Simon & Schuster, 1996; I.R. Hardy, ''The Evolution of ARPANET Email,'' Univ. California, Berkeley, master's thesis, 1996; J. Abbate, *Inventing the Internet*, MIT Press, 1999. Quarterman's book sought to document the state of networking in the world in 1990 and is a tremendously valuable testament to a time just before the Internet took over. The works of Salus, Hafner and Lyon, and Abbate are general histories in which email plays a modest part—this article is, in some sense, the fine-grained version of email's history. Hardy's thesis seeks to understand the social dynamics of the community in which email developed and is a useful complement to this work. T. Haigh, ''The Web's Missing Links: Search Engines and Portals,'' *The Internet and American Business*, W. Aspray and P. Ceruzzi, eds., MIT Press, 2008, also has a useful perspective.
2. J.K. Reynolds, *Post Office Protocol*, Internet Request for Comments No. 918, Oct. 1984; J. Myers and M. Rose, *Post Office Protocol—Version* 3, Internet Request for Comments No. 1939, May 1996. (For information on retrieving RFCs online, see Ref. 5).
3. M.R. Crispin, *Interactive Mail Access Protocol: Version 2*, Internet Request for Comments No. 1064, July 1988.
4. T. Van Vleck, ''The History of Electronic Mail,'' http://www.multicians.org/thvv/mail-history.html.
5. The Internet Request for Comments series is maintained online. To retrieve a particular RFC, use http://www.rfc-editor.org/rfc/rfc#.txt, where # is replaced with the one-, two-, three-, or four-digit RFC number.
6. R.W. Watson, *A Mail Box Protocol*, Internet Request for Comments No. 196, 20 July 1971.
7. R. Tomlinson, personal communication, 13 Apr. 2006.
8. A similar line of thinking had led to email in Multics. Louis Pouzin wanted a way to send a message to an operator and that idea morphed into sending messages between users. J. Klensin, personal communication, 10 June 2006.
9. D.G. Bobrow et al., ''TENEX, a Paged Timesharing System for the PDP-10,'' *Comm. ACM*, vol. 15, no. 3, 1972.
10. SNDMSG's origins are uncertain. Tomlinson ported SNDMSG to TENEX from another operating system. He believed the operating system was Berkeley's SDS-940 system, but the SDS-940 veterans report it did not have an email program.
11. The choice of @ did cause some controversy. It turned out that @ was a reserved character in Multics that caused all input to that point on a line to be deleted.
12. A. Bhushan, *File Transfer Protocol*, Internet Request for Comments No. 114, 10 Apr. 1971; A. Bhushan et al., *The File Transfer Protocol,* Internet Request for Comments No. 265, 17 Nov. 1971.
13. A side note: the problem of developing a general distributed file system (which was the goal of the initial FTP work) turned out to be excruciatingly hard and was not solved until the early 1980s and required the development of the concept of remote procedure call (see B.J. Nelson, ''Remote Procedure Call,'' doctoral dissertation, Carnegie Mellon Univ., 1981) and distributed transactions (W.E. Weihl, ''Transaction Processing Techniques,'' *Distributed Systems*, 2nd ed., S. Mullender, ed., Addison-Wesley, 1993).
14. A.K. Bhushan, *Data and File Transfer Workshop Notes*, Internet Request for Comments No. 327, 27 Apr. 1972.
15. A.K. Bhushan, *File Transfer Protocol*, Internet Request for Comments No. 354, 8 July 1972.
16. A.K. Bhushan, *Comments on the File Transfer Protocol*, Internet Request for Comments No. 385, 18 Aug. 1972. Despite the title, it is the document that defined MLFL and MAIL and appears to have been the standard reference for the next eight years.
17. D. Crocker, personal communication, 26 Apr. 2006.
18. The Arpanet community eventually developed a term for this kind of problem: the n × m rule. The n × m rule, paraphrased, says that one should abhor designing systems where the consequence of adding a new system is that every other system needs to learn how the new system works (i.e., where the new system places users' mailboxes).
19. A.K. Bhushan, RFC-385; M. Padlipsky, personal communication, 14 Apr. 2006.
20. M. Padlipsky, ''And They Argued All Night …,'' note at http://www.lafn.org/%7Eba213/allnight.html.
21. S. Lukasik, oral history interview by J.E. O'Neill, 17 Oct. 1991, Redondo Beach, Calif., OH 232, Charles Babbage Inst. Lukasik guessed he was using email on Arpanet by 1971 (p. 11). That is too early, but 1972 is plausible.
22. The original name was Tape Editor and COrrector, but by 1973, the acronym had evolved. Thanks to Dan Murphy (TECO's author) and John Vittal for tracking down when the name changed.
23. Lukasik (Ref. 21) says Roberts produced RD overnight. Roberts dates the invention of RD to

July 1972 in his Internet chronology (http://www.packet.cc/internet.html).

24. M. Yonke, personal communication, 26 Apr. 2006. There was an intermediate program between NRD and BANANARD, called WRD (for Wessler's RD). Yonke recalls it was only around briefly and was largely Wessler's code with bug fixes, but otherwise unmodified.
25. M. Yonke, personal communication, 26 Apr. 2006; S. Crocker, personal communication, 31 May 2006; J. Vittal, personal communication, 5 June 2006; Yonke remembers the index size as 5,000 while Crocker remembers it as a much smaller number (a few hundred). Vittal remembers hitting the limit.
26. B.D. Wessler, personal communication, 17 Apr. 2006.
27. R. Tomlinson, personal communication, 13 Apr. 2006; J. Vittal, personal communication, 5 June 2006.
28. The meeting announcement (A. McKenzie, *File Transfer Protocol—Meeting Announcement and a New Proposed Document*, Internet Request for Comments No. 454, 16 Feb. 1973) contains a draft new specification and includes suggested improvements to MAIL and MLFL.
29. J. Day, personal communications, 14 and 16 Apr. 2006.
30. N. Neigus, *File Transfer Protocol*, Internet Request for Comments No. 542, 12 Aug. 1973.
31. The common set of attendees was Abhay Bhushan (from MIT), Bob Braden (UCLA), Alex McKenzie (BBN), Jon Postel (UCLA), and Jim White (SRI).
32. That there were FTP meetings at roughly the same time of year in both 1972 and 1973 has caused some confusion decades later. People are sometimes confused about which meeting they attended.
33. J. Postel, ''Mail Protocol,'' DDN NIC memo 29588, 18 Feb. 1976 in *ARPANET Protocol Handbook,* DDN Network Information Center, Jan. 1978.
34. Allowing multiple addresses in a MLFL/MAIL command was proposed in A. Bhushan, *File Transfer Protocol (FTP) Status and Further Comments*, Internet Request for Comments No. 414, 29 Nov. 1972, and again (now using new commands) in K. Harrenstein, *FTP Extension: XRSQ/XRCP*, Internet Request for Comments No. 743, 30 Dec. 1977. NIC 29588 suggests that some sites used the RFC-414 scheme but that it was not universally accepted.
35. There were some proposals for an email protocol (the preface to RFC-724 mentions ''Several versions of such a protocol have been proposed ...''). However, none seem to have gotten serious attention; only one seems to have been issued as an RFC (J.E. White, *Proposed Mail Protocol*, Internet Request for Comments No. 524, June 1973), and there's a strong impression that the community paid very little attention to the issue.
36. J. Vittal, ''MSG—A Simple Message System,'' *Computer Messaging Systems*, R.P. Uhlig, ed., North Holland, 1981.
37. BBN collected Arpanet traffic measurements monthly during this time, but I have been unable to find a copy of them and thus could not verify this claim.
38. S. Walker, ''Message Group Status,'' email to MsgGroup of 7 June 1975.
39. The Navy was a pioneer in the use of email for operational needs. In 1973, the Navy had inaugurated the ''Navy Communications Processing and Routing System (NAVCOMPARS),'' an internal email system to distribute orders to shore bases and to ships (messages to ships were relayed via shortwave radio using human operators). NAVCOMPARS remained a key part of the Navy's infrastructure until it was turned off in 2002. B.M. Hintz, ''The Naval Communications Processing and Routing System: Analysis of an Automated System,'' master's thesis, Naval Postgraduate School, Mar. 1976. A good description of the system as it was operating in 1984 can be found in S. Blumenthal et al., ''NAVCAMS LAN Engineering Plan,'' BBN Report 5907, Mar. 1985.
40. S. Walker, personal communication, 8 June 2006.
41. D.P. Deutsch and D.W. Dodds, *Hermes System Overview*, BBN Report 4115, May 1979.
42. D.H. Crocker, *Framework and Functions of the ''MS'' Personal Message System*, tech. report R-2134-ARPA, RAND Corp., Dec. 1977.
43. A copy of the original memo can be found at http://rand-mh.sourceforge.net/book/overall/hiofmh.html.
44. D.M. Ritchie and K. Tompson, ''The UNIX Time-Sharing System,'' *The Bell System Technical J*, vol. 57, no. 6, July-Aug. 1978, part 2, pp. 1905-1930.
45. Dave Crocker observes the interesting counterpoint that attempts to ''enhance'' MH, such as xmh and mhe, have sought to move the MH commands back into a monolithic program.
46. Dave Crocker reports that the MS manual, to his surprise, does not describe features for searching. D. Crocker, personal communication, June 2006.
47. I interacted with Rose and Jacobson on MH support in the 1980s. Other names come from J. Peek, *MH & xmh: Email for Users & Programmers*, O'Reilly and Associates, 3rd ed., 1995.
48. A. Bhushan et al., *Standardizing Network Mail Headers*, Internet Request for Comments No. 561, 5 Sept. 1973.

49. T.H. Myer and D.A. Henderson, *Message Transmission Protocol*, Internet Request for Comments No. 680, Apr. 1975.
50. See E. Stefferud, ''MSGGROUP Situation Report #1,'' email to Header-People of 2 Dec. 1975.
51. In particular, Mooers wrote emails on behalf of the BBN team explaining details of RFC-680. Years later, Mooers was CSnet's ''postmistress'' and internationally known for her expertise in solving email problems.
52. J. Haverty, ''Re: [ih] NIC 7104 (ARPANET Protocol Handbook)'' email to Internet-History mailing list of 28 Apr. 2006.
53. See RFC-724 preface.
54. Interview with A. Vezza on 3 May 2006. He believes his (now lost) memo, ''Message Services Committee Minority Report,'' Jan. 1975, expressed the view that headers should be machine readable. See also the preface to RFC-724, which reflects the continued debate.
55. The initial membership of the new committee was Walker, John Seely-Brown, David Farber, Ken Pogran, and John Vittal. J. Vittal, personal communication, 5 June 2006.
56. For key notes in the discussion, see the note from J. Haverty (JFH @ MIT) on 30 Sept. 1976, ''Re: your message to MSGGROUP at from and sender …,'' and J. Vittal, ''Some comments (RFC-724, etc.) …'' of 9 Nov. 1976, both emails to MsgGroup. Confusing the discussion is that an early draft of RFC-724 was apparently distributed, with its assigned RFC number, in 1976 (well before its official publication date). K. Pogran et al., *Proposed Official Standard for the Format of ARPA Network Messages*, Internet Request for Comments No. 724, 12 May 1977.
57. J. Vittal, ''Comments on the state of the world,'' email to Header-People mailing list of 29 Oct. 1977.
58. See the numerous emails between 4 and 11 Oct. 1977 in Header-People.
59. Mailing lists had an odd history in the message format standardization process. No later than 1975, there were mailing lists as we know them today, in which email to, say, *MsgGroup@ISI* was delivered to host ISI. Host ISI then redistributed the message to the members of the list. Yet there seem to have been a number of different practices intended to show that the message was to a mailing list. So, at one time, ISI would rewrite the outbound TO field of MsgGroup messages to read ''[ISI]MSGGROUP:''.
60. D. Crocker, *Standard for the Format of ARPA Internet Text Messages*, Internet Request for Comments No. 822, Aug. 1982; P. Resnick, *Internet Message Format*, Internet Request for Comments No. 2822, Apr. 2001.
61. CSnet supported RFC-733 format from the start. The UUCP network took somewhat longer. The driving forces were *sendmail* (used on many UUCP systems) and *netnews B*, which intentionally used 733 format for bulletin boards. Both software systems (plus a desire to easily gateway to the Internet) pushed the community to informally standardize on 733 for email. (S. Bellovin, personal communication, 30 May 2006; M. Horton, personal communication, 6 June 2006; see also, M. Horton, *UUCP Mail Interchange Format Standard*, Internet Request for Comments No. 976, Feb. 1986.) Bitnet started using a custom VM email format but soon shifted to 733 (J. Klensin, personal communication, 26 May 2006).
62. A. Bhushan, *File Transfer Protocol (FTP) Status and Further Comments*, Internet Request for Comments No. 414, 29 Nov. 1972, lists the implementation status of various FTP implementations and observes that Clements has implemented email retransmission.
63. See Deutsch and Dobbs, *Hermes System Overview*, p. 21.
64. Compare with B. Reid, ''Let's hear it for uniform standards,'' email to Header-People of 10 Feb. 1978.
65. This decision to interconnect is, in retrospect, somewhat surprising. The different networks did view themselves as competing with each other. However, they also viewed themselves as competing with the postal services (derisively dubbed ''snail mail'') and prided themselves on getting email where it belonged faster and more effectively than paper-based mail. Credit should also be given to Larry Landweber, a member of both CSnet's and Bitnet's boards, and a vigorous advocate of interconnecting networks.
66. D.A. Nowitz and M.E. Lesk, ''A Dial-Up Network of UNIX™ Systems,'' 18 Apr. 1978, *Unix Programmer's Manual*, 7th ed., Bell Telephone Laboratories, 1979.
67. J.S. Quarterman, *The Matrix*, p. 251 and 278.
68. P. Honeyman and S. Bellovin, ''PATHALIAS or the Care and Feeding of Relative Addresses,'' *Proc. 1987 Summer Usenix Conf.*, 1986, Usenix Assoc., pp. 126-141.
69. D. Comer, ''The Computer Science Research Network CSNET: A History and Status Report,'' *Comm. ACM*, vol. 26, no. 10, 1983, pp. 747-753.
70. Peter Denning, one of the CSnet principals, remembers that, to make the CSnet proposal ''researchy'' enough to be acceptable to the National Science Board, the proposal emphasized ''resource sharing'' rather than email, but everyone, including the junior NSF staffers, understood this was a fig leaf for email. (P. Denning, personal communication, 5 June 2006.)

71. D.E. Comer and J.T. Korb, ''CSNET Protocol Software: The IP-to-X.25 Interface,'' *Proc. ACM SIGCOMM '83*, ACM Press, 1983, pp. 154-159.
72. L. Lanzillo and C. Partridge, ''Implementation of Dial-up IP for UNIX Systems,'' *Proc. 1989 Winter Usenix Conf.*, Usenix Assoc., pp. 201-208.
73. Surviving source code (version 2.7 from 1981) is about 6,800 lines of C code. There seem to have been no technical papers describing *delivermail*. The discussion here comes primarily from reading the source code and its Unix manual pages.
74. D.H. Crocker, E.S. Szurkowski,, and D.J. Farber, ''An Internetwork Memo Distribution Capability—MMDF,'' *Proc. 6th ACM/IEEE Data Comm. Symp.*, ACM Press, 1979, pp. 18-25.
75. This list is a subset of the list of differences in E. Allman, ''Mail Systems and Addressing in 4.2bsd,'' *Proc. 1983 Winter Usenix Conf.*, Usenix Assoc., 1983, pp. 53-62.
76. E. Allman and M. Amos, ''Sendmail Revisited,'' *Proc. 1985 Summer Usenix Conf.*, Usenix Assoc., 1985, pp. 547-555.
77. See J. Postel, ''Internet Meeting notes—4, 5, & 6 February 1980,'' *Internet Engineering Notes,* no. 134, 29 Feb. 1980, which notes that ISI was working on ''Internet Mail, which includes the development of mechanisms for delivery of mail in an internet and provision for multi-media data in the mail.''
78. J. Postel, ''Internet Meeting Notes—14 & 15 May 1980,'' *Internet Engineering Notes,* no. 145, 25 May 1980.
79. V. Cerf and J. Postel, *Mail Transition Plan*, Internet Request for Comments No. 771, Sept. 1980.
80. S. Sluizer and J. Postel, *Mail Transfer Protocol*, Internet Request for Comments No. 772, Sept. 1980. Some people have conflated MTP and MP (the Mail Protocol—see the Early Multimedia section) and incorrectly believe that MTP supported multimedia.
81. RFC-771 does not explain why it was written. But RFC-772 makes clear that MTP is intended solely for use for gateways.
82. Suzanne Sluizer recalls Jon ''Postel saying people thought that MTP was too complicated.'' C.J. Bennett, ''A Simple NIFTP-Based Mail System,'' *Internet Engineering Notes*, no. 169, 23 Jan. 1981, lists issues with MTP on pp. 4 and 5.
83. J. Postel, ''Internet Meeting Notes—28-29-30 January 1981,'' Internet Engineering Notes, no. 175, 13 Mar. 1981, lists four implementations: MIT, ISI, DCEC, and COMSAT.
84. S. Sluizer and J. Postel, *Mail Transfer Protocol: ISI TOPS-20 Implementation*, Internet Request for Comments No. 784, 1 July 1981; S. Sluizer and J. Postel, *Mail Transfer Protocol: ISI TOPS-20 File Definitions*, Internet Request for Comments No. 785, 1 July 1981.
85. E. Stefferud, ''Subdivision of Messages,'' email to MsgGroup of 11 July 1975.
86. The first use of envelope to mean metadata appears to be by D. Crocker, E. Szurkowski, and D. Farber, ''An Internetwork Memo Distribution Capability—MMDF,'' *Proc. 6th ACM/IEEE Data Comm. Symp.*, ACM Press, 1979, pp. 18-25.
87. The idea for pipelining originated with Phil Karn around 1990, when he told it to me, and I in turn, repeated the idea to Van Jacobson. Jacobson thought it was a wonderful idea, and put it into *sendmail* only to discover that many SMTP implementations failed if pipelining was turned on. Some of the painful experience is described in P. Karn, email to IETF mailing list of 8 Sept. 1993. Eventually, the IETF approved a pipelining extension to SMTP to make it official: N. Freed, *SMTP Service Extension for Command Pipelining*, Internet Request for Comments No. 2920, Sept. 2000.
88. C. Partridge, *Duplicate Messages and SMTP*, Internet Request for Comments No. 1047, 1 Feb. 1988.
89. Unfortunately, SMTP is no longer quite this simple. Some commands now have additional parameters (a consequence of the 8-bit enhancements; J. Klensin, *Simple Mail Transfer Protocol*, Internet Request for Comments No. 2821, Apr. 2001), and there's pressure to standardize the error codes to avoid dependence on the error message, which may be a local language (J. Klensin, personal communication, 11 June 2006).
90. E. Feinler et al., *DoD Internet Host Table Specification*, Internet Request for Comments No. 810, 1 Mar. 1982.
91. Z. Su and J. Postel, *Domain Naming Convention for Internet User Applications*, Internet Request for Comments No. 819, 1 Aug. *1982*.
92. See A.D. Birrell et al., ''Grapevine: An Exercise in Distributed Computing,'' *Comm. ACM*, vol. 25, no. 4, 1982, pp. 260-274.
93. P. Mockapetris, *Domain Names: Concepts and Facilities*, Internet Request for Comments No. 882, 1 Nov. 1983; P. Mockapetris, *Domain Names: Implementation Specification*, Internet Request for Comments No. 883, 1 Nov. 1983.
94. P. Mockapetris and K.J. Dunlap, ''Development of the Domain Name System,'' *Proc. ACM SIGCOMM '88*, ACM Press, 1988, pp. 123-133.
95. C. Partridge, ''MF in domain database,'' message to NameDroppers mailing list of 29 Oct. 1985.
96. C. Partridge, ''MD and MF for one host,'' message to NameDroppers mailing list of 11 Nov. 1985.
97. The description of the issues is now lost, but it seems useful to reconstruct it from memory. If a name could have both MD and MF records associated with it, we needed a set of rules for delivery in the

presence of both records. The obvious answer was that a host that was neither an MD nor an MF for the name could deliver email to either the MD or the MF; a host that was an MF could only deliver to an MD; and a host that was an MD could do a DNS lookup, but, once it realized it was an MD, had to look in other databases to figure out how to deliver the message. Now consider the problem of host H, trying to deliver a message to domain name D. H looks up D in the DNS and gets back a set of email resource records. H must then examine all the records to see if H is listed as either an MD or an MF for D to behave in accordance with the delivery rules. Herein lay a problem. It was possible in the DNS to deliver an incomplete list of MDs and MFs. In particular, the DNS's caching mechanism (combined with the fact that one could query for MDs and MFs either individually or using an aggregate MAILA query) allowed for look up responses to contain either the MDs for D or the MFs for D, or both. And if H did not get both MDs and MFs, H could make an incorrect decision.

98. These discussions involved both private and public messages. The private messages have been lost. The key public messages are P. Milazzo, ''Re: MD and MF for one host,'' message to NameDroppers on 11 Nov. 1985 in response to the message cited in Ref. 95; C. Partridge, ''Mailers use MD and MF,'' message to NameDroppers on 12 Nov. 1985; and P. Mockapetris, ''MD, MF and larger issues,'' message to NameDroppers on 15 Nov. 1985. It is my recollection that the Mockapetris note came after a private email exchange among Postel, Mockapetris, and Partridge. Rudy Nedved (then of CMU) and Jon Crowcroft (then of University College London) also made important contributions, especially in thinking how MX RRs interacted with sites that gatewayed email.
99. P. Mockapetris, *Domain Systems Changes and Observations*, Internet Request for Comments No. 973, Jan. 1986; C. Partridge, *Mail Routing and the Domain System*, Internet Request for Comments No. 974, Jan. 1986. While the RFCs were issued in late January, I recall that the work on RFC-974 was largely done by Thanksgiving of 1985 (which is remarkable, given the issues with MD and MF surfaced on 11 Nov.) and the delay until January was to give Mockapetris time to think about how to address other DNS issues so that RFC-973 could be a comprehensive update.
100. C. Partridge, ''Mail Routing Using Domain Names: An Informal Tour,'' *Proc. 1986 Summer Usenix Conf.*, Usenix. Assoc., 1986, pp. 366-376.
101. No notes of this meeting survive. The attendance list is crafted from my recollections and those of Mary Ann Horton. There's uncertainty about whether Steve Kille was there (he's not sure) but it is more likely than not. Kille would have provided an international perspective (he was at University College London) and an X.400 perspective. The account of the meeting is my recollection. Horton's recollections place somewhat more emphasis on finalizing the list of top-level domains.
102. The issue surrounding .net was that SRI (operator of the DDN NIC) and BBN (operator of the CSnet CIC) competed for network operations contracts and differed in their strategies. BBN's approach was to build the brand of the entity for which BBN operated the network (so, for instance, BBNers on the CSnet project had CSnet business cards) on the theory that BBN's dedication to building the brand made BBN more attractive to the customer. SRI sought to strongly link the entity to SRI, on the theory the customer would be more reluctant to change operators. So CSnet wanted to see a *.net* top-level domain so that NIC's name would be, say, *nic.inter.net*. SRI wanted the NIC's name to be *nic.sri.com*. In the event, .net was created, but the NIC became *nic.ddn.mil.*
103. For Postel's views on *.arpa* and *.uucp* and the like, see his email ''re: naming and routing'' to the NameDroppers mailing list on 8 Feb. 1995.
104. R.E. Millstein, ''The National Software Works: A Distributed Processing System,'' *Proc. ACM'77 Conf.*, ACM Press, pp. 44-52.
105. J. Postel, *Internet Message Protocol*, Internet Request for Comments No. 753, Mar. 1979.
106. The *uuencode* format was also adopted by several early email tools, notably Microsoft Mail and Lotus cc:Mail, for packaging attachments. M.A. Horton, personal communication, 6 June 2006.
107. J. Postel, *Internet Message Protocol*, Internet Request for Comments No. 759, Aug. 1980. J. Postel, *A Structured Format for Transmission of Multi-Media Documents*, Internet Request for Comments No. 767, Aug. 1980.
108. V. Cerf, *Comments on NCP/TCP Mail Service Transition Strategy*, Internet Request for Comments No. 773, 1 Oct. 1980.
109. For a brief overview of the projects, see J. Postel, *Multimedia Mail Meeting Notes*, Internet Request for Comments No. 807, 9 Feb. 1982.
110. R.H. Thomas et al., ''Diamond: A Multimedia Message System Built on a Distributed Architecture,'' *Computer*, Dec. 1985, pp. 65-78.
111. Concurrently a similar project, Project Athena, was under way at MIT.
112. The first paper on the Andrew Messaging System cites none of the prior Internet-based work. See J. Rosenberg, C.F. Everhart, and N.S. Borenstein, ''An Overview of the Andrew Message System: A Portable, Distributed System for Multi-media

Electronic Communication,'' *Proc. ACM SIGCOMM* '87, ACM Press, pp. 99-108.

113. For a general overview of work in data encoding for the period, see C. Partridge and M. Rose, ''A Comparison of External Data Formats,'' *Message Handling Systems and Distributed Applications (Proc. IFIP Workshop on Message Handling)*, E. Stefferud and O. Jacobsen, eds., North Holland, 1989.
114. There were some pioneers. The earliest idea I have seen for an external data format is J. Haverty, *MSDTP-Message Services Data Transmission Protocol*, Internet Request for Comments No. 713, 6 Apr. 1976. It contains a remarkably thorough understanding of the problem of creating an external data format. Haverty was at BBN when Deutsch and Vittal were doing their work.
115. D. Deutsch, R. Resnick,, and J. Vittal, *Specification of a Draft Message Format Standard*, BBN Report 4486, Sept. 1980. D. Deutsch et al., *Specification for Message Format for Computer Based Message Systems (Revised)*, BBN Report 4765R, 23 Apr. 1982. Online literature generally gives all credit for ASN.1 to Jim White. As explained to me (some years ago), White gets credit for ASN.1's language for expressing types, but the actual on-the-wire encoding (the Basic Encoding Rules) is the creation of Deutsch's group. The dates of these two BBN reports are consistent with that story (they define what clearly became the encoding rules) and explain why the encoding rules are completely unlike Courier, which was White's invention and the Xerox external data format of the time.
116. There is considerable debate, even today, about how viable X.400 would have been as the Internet's email system. Several readers of the paper felt the characterization of X.400 both in terms of the quality of its technology and its chances for success is far too generous. Some others felt this was about right.
117. *Proc. Nineteenth IETF Meeting*, IETF, Dec 1990, pp. 72-76.
118. *Proc. Twentieth IETF Meeting*, IETF, 11-15 Mar. 1991, M. Davies and G. Vaudreuil, eds., pp. 75-84.
119. There are some hints in the meeting notes that the two groups initially may have been confused about whether they were complementary or competing. Participants' memories vary. But it is hard to believe that there was much competition, given they had the same chairman.
120. The decision not to support *uuencode* is noted on p. 62 of *Proc. Twenty-Second IETF Meeting*, IETF, 18-22 Nov. 1991, M. Davies, C. Clark, and D. Legare, eds.
121. Until Nov. 1991, both the SMTP and RFC-822 extensions groups were chaired by Greg Vaudreuil. Vaudreuil was a good group leader, but also new to the field and quite young and thus, unlike Klensin, in no position to dictate a solution to a fractious (and senior) group of techies. Vaudreuil continued as chair of the 822-extensions group and brought it to a successful conclusion. Klensin remembers Phill Gross, IETF chair, ''dragged me, I'm tempted to say kicking and screaming'' into taking on the SMTP problems (J. Klensin, personal communication, 11 June 2006).
122. M. Davies, ed., *Proc. Twenty-Third IETF Meeting*, IETF, 15-20 Mar. 1992.
123. Recollections differ slightly about how this change of course came about. (J. Klensin, personal communication, 11 June 2006; D. Crocker, personal communication, 31 May 2006). Rose pushed for the simplification. What is unclear is whether EHLO was Rose's idea, or the reworking of an earlier idea by Klensin that the working group had discarded. I cannot find documentation pointing to one or the other answer.
124. J. Klensin et al., *SMTP Service Extension for 8bit-MIME Transport*, Internet Request for Comments No. 1426, Feb. 1993.
125. Of the Dec. 1990 group, only Bob Braden of ISI had written an MTA or participated in crafting an email standards document.
126. Borenstein, Freed, Crocker, and Klensin's background is described earlier in the article. Stefferud ran the Msg-People mailing list in the 1970s and coined the term ''envelope.'' Fair was widely respected as an expert at keeping email systems running and, at the time, managed Apple Computer's email systems. Huitema was a pioneer in networking in France.

Craig Partridge is chief scientist for networking at BBN Technologies, where he has worked on data networking problems since 1983. He is best known for his work on email routing, TCP round-trip time estimation, and high-performance router design. He received an MSc and a PhD, both in computer science, from Harvard University. Craig is the former editor in chief of *IEEE Network Magazine* and *ACM Computer Communication Review* and is an IEEE Fellow.

Readers may contact Craig Partridge about this article at craig@bbn.com.