Netnews: The Origin Story

Steven M. Bellovin
Department of Computer Science
Columbia University
https://www.cs.columbia.edu/~smb

June 27, 2023

Abstract

Netnews, sometimes called Usenet, was arguably the first social network It had a profound influence on online socializing, including helping to give to the world the current slang meanings of words like "spam", "troll", and "flame". It was where many technologies we now take for granted were first announced, including Linux, the World Wide Web, and the graphical web browser. But its design was a function both its design goals and the technological context of the time. I describe those and a variety of other early design decisions, those which were right, those which were wrong, and those which were inevitable.

1 Introduction

Netnews, sometimes called Usenet,* was, for a time, the only way in which most people could share information with others: technical queries and solutions, discussions of child-rearing, debates about politics, and more. It was in principle a fully decentralized network, but in practice—and by intent at the beginning—was more a series of linked star networks. But why did it take the shape that it did? Some decisions were forced by the technology of the time, some others were arbitrary, and some were due to the relative ignorance of its creators: none of us were specialists in networking, human interaction, cryptography, and more.

At its peak, Netnews was *the* forum where major announcements appeared. Linus Torvalds used it to announce Linux; Tim Berners-Lee used it to announce the World Wide Web, and, ironically, it was where Marc Andreessen announced Mosaic, the first

*Strictly speaking, "Netnews" is the name of the technology. "Usenet" was a particular instantiation, at one point defined as "the set of machines that receive the newsgroup NET.general." Conceptually, there could have been completely disjoint sets of computers running Netnews technology, e.g., for intra-company use. In practice, separate newsgroups served that purpose.

Jim Ellis coined the name "Usenet" as a riff on "Usenix." Shortly before we created Netnews, the then-Unix Users Group was forced to change its name by Bell Labs' trademark lawyers. They adopted the name "Usenix"; we said explicitly that we hoped that Usenix would "take an active (indeed central) role in the network."

graphical web browser, arguably the technology that did the most to make Netnews obsolescent.

As a major cultural force, Netnews fell into eclipse by the late 1990s. The advent of AOL as a more available alternative for ordinary users, Web-based chatrooms, and the later rise of social networks such as Friendster and MySpace (and of course somewhat later Facebook and Twitter) led to its decline: people could find the content and the communication they needed in other forms, forms that were often much more user-friendly. And to some extent, Netnews was a victim of its own success: it was carrying so much traffic that most sites could no longer handle the load, nor could users keep up with the volume. More focused fora gained ground.

All that said, even in retrospect most of the early decisions were correct, save for one: the designers never planned sufficiently for success. As a then-graduate student, I was one of the original designers and the first implementor of Netnews; this is its early history as I remember it.

1.1 What is Netnews?

Netnews is a distributed bulletin board system. It is composed of multiple sections, called "newsgroups;" these can be topic-oriented, e.g., sci.crypt for discussing cryptography, or geographic: tri.used-cars might be for posting car ads within the Research Triangle area of North Carolina.

Users create "posts"; these are relayed to neighboring machines via a flooding algorithm. Duplicate detection is done by the receiving system, though there is some ability to prevent forwarding loops. Replies to posts can be either new posts or direct email messages.

In the early days (and always in the original design), transmission was via dial-up modem; later on, direct Internet transmission became possible.

2 Technological and Economic Background

The original design of Netnews dates to the fall of 1979. Most computing power was concentrated in large mainframes. Some academic departments had so-called minicomputers such as the DEC (Digital Equipment Corporation) PDP-11 or the later VAX-11, but these were all at best timesharing machines. Users connected to them via dumb terminals, generally at speeds no greater than 9600 bps. Dial-up, if available at all, was generally limited to 300 bps; 1200 bps modems were just entering service. IBM had a line of high-speed 3270 terminals, but these were generally connected via coaxial cable within a building.

Personally owned computers—microcomputers, in the terminology of the day—were rare and were the domain of a few hobbyists. Most were very small and generally lacked hard drives; bulk storage was via audio cassette tape or (for the lucky few) on floppy disks with a capacity of about 1.5 megabytes.

This was also an era before most inter-vendor networking; such interconnections as did exist tended to use vendor-proprietary hardware and protocols. To be sure, there

were production networks,[23] and Ethernet had been invented,[15] but few sites had access to such and their reach was not very great.

The notable exception to the networking story was, of course, the ARPANET. It was explicitly a long-haul, vendor-independent network. Maps from that era show connections to computers made by DEC, IBM, Univac, Honeywell, ICL, and more. However, there was a catch: to be on the ARPANET, you needed to be part of the military, be a defense contractor, or have a Department of Defense research contract. The IMPs—Interface Message Processors, the analog to today's routers—were actually minicomputers in their own right.

Even modems were comparatively unusual. The reason was partly technological and partly cost, though you could buy a high-quality 300 bps acoustic coupler—a device that had a microphone and a speaker that used sound to connect a telephone handset to a jack that could be connected to a computer's serial port—for a few hundred dollars. The bigger issue was regulatory: generally speaking, only phone company-owned equipment could be hard-wired to the telephone network. AT&T, the dominant phone company at the time in the U.S., would only lease modems and the like, which meant that even offering external connections incurred a continuing monthly cost.

There was one further technological issue, this time a positive one: the Unix operating system from Bell Labs. It ran quite nicely on PDP-11 minicomputers. By today's standards, the PDP-11 was a very limited machine—it had a 16-bit address space, and although a computer could have more physical memory than 64K bytes, no more than that was addressable at any one time. (In fact, the PDP-11 allowed programs to have separate 16-bit address spaces for instructions and data. A number of standard, large programs took advantage of that.) Duke University's computer science department had a PDP-11/70; my graduate computer science department, at the University of North Carolina at Chapel Hill, had a smaller, slower, but nevertheless still capable PDP-11/45. Furthermore, Seventh Edition Unix came with a communications package known as `uucp`, which ran on dial-up links and provided file copy and remote execution facilities. `Uucp` was the eventual technical underpinning of Netnews.

3 Problem Definition

Although 7th Edition Unix was released in early 1979, conversion was not a simple matter. Apart from anything else, early Unix distributions were primarily in source code form, which led to a thriving culture of shared modifications to the code base. When upgrading to a new version of Unix, each of these modifications needed to be ported, replaced, or abandoned as not worth the effort. One useful change to the `login` command was some code to display a message once, and once only, to a user who logged in. There was a standard system facility to display a login-time message, but at 15 characters per second, the speed of some of the hardcopy terminals in use, printing a longer message at every login was undesirable. This change to the base system was seen as desirable, but it wasn't straightforward to carry forward; the login command had changed dramatically between 6th Edition and 7th Edition Unix. Besides, there was a desire for more functionality.

Two Duke grad students, the late Jim Ellis[6] and Tom Truscott, had grander goals.

They conceived of not just a system to display local administrative messages, but rather a networked system, one where a message created on one system could be seen on others. This was partly a response to the computing environment at Duke University, where there were at least two other PDP-11s running Unix, and partly a desire to tie together a broader community. At a minimum, we wanted a system that could be used to, for example, announce a talk at one local university and have the announcement seen at others in the area. But there was never an intent to restrict the network to "official" use; from the very start, we wanted to support things like used car ads that would reach the entire area.

Ellis and Truscott called a meeting at Duke University to discuss the problem; I was invited to attend the meeting. We fleshed out the goals, made some preliminary engineering estimates, designed a first cut at the protocol, and discussed the connectivity problem. With all that settled, the meeting broke up; I returned to Chapel Hill to build the first prototype.

3.1 Home-Built Auto-Dialers

Per the discussion in Section 2, neither UNC nor Duke had "official" autodialers. Modems with built-in autodialers were still several years in the future. As a student-run skunkworks project we couldn't simply buy a DEC DN11 interface and lease a Bell 801 autodialer, so we had to design around the problem. Duke implemented a solution first; inspired by their design, I came up with my own, which was implemented by the department's excellent electronics techician. In both cases, we used acoustic couplers that had phone handsets permanently inserted into them.

A normal phone circuit of the time was two-wire, with a potential of 48 volts across those wires. When someone lifted the handset, the phone went "off-hook", lowering the voltage to 3–9V. This drop was recognized by the central office, which would then listen for dialing instructions (or complete a call, for inbound calls). Crucially, an on-hook phone presented essentially infinite DC resistance; this mean that an open line was indistinguishable from an on-hook phone. We took advantage of that by connecting a normally open relay across one of the wires. We then used serial port control signals to activate that relay.

The standard for serial ports, RS-232, was hideously complex, but most of the complexity is irrelevant to what we did. Two of the control lines, Data Terminal Ready (DTR) and Carrier Detect (CD), did the work. When a program wanted to use a serial port, the device driver raised the DTR signal; this caused the relay to close, and hence the phone to be perceived as off-hook. (The handset, as noted, was already removed from the base and inserted into the acoustic coupler.)

The next problem was dialing. Pulse dialing, for rotary dial phones, was still extremely common at the time; in fact, virtually all telephones in Chapel Hill were rotary dial. Pulse dialing worked by going back on-hook briefly, at a specified rate; a brief interruption of the circuit was seen as a dial pulse, not as the phone hanging up.[11] Dialing the digit "1" took a single pulse, a "2" was two pulses, etc. The standard called for a 3:2 break:make ratio over the course of one second. That is, the phone was effectively on-hook for .06 seconds and back off-hook for .04 seconds for each pulse. A longer off-hook interval separated individual digits.

The timing standard, though, was fairly loose—it had to be, since phone dials were mechanical and could be affected by wear, dirt, temperature, and more. We used software to control the timing of the relay going back on-hook. The timer resolution of the Unix systems of the day was 60 Hz; this let us implement a 2:1 make-break ratio, which was adequate. I wrote a device driver that emulated the standard DN11 driver, permitting seamless interface to `uucp`. When our modem detected a carrier signal from the far end, it raised the CD signal; this let the open request to the serial port complete. (A year or two later, we obtained a 1200 bps modem, which was hardwired to the phone line; it required a somewhat more complex signaling interface to the computer, but one that was well within the abilities of our electronics technician, and the dialing interface remained the same.)

The scheme may not have been elegant, and it certainly violated phone company tariffs, but it worked.

3.2 A Star Topology

The autodialer problem was not the only economic issue that had to be solved. In 1979, "long distance phone calls"—phone calls outside the local calling area—were expensive, with costs varying with distance and time of day. Both UNC and Duke had the technical ability to call remote sites, but we did not want to pay for many such calls. Dialing a local call was free, which was advantageous at Duke, since it let the CS department feed the other two Unix machines, but it would not work for the larger network we envisioned.

The solution was simple: Duke would poll any site that wished, as frequently as they wished, on one condition: that site would reimburse Duke for the phone calls.

There is an important corollary to this idea, one that speaks well of the Duke computer science department at the time. There was no way that a group of graduate students could have done billing, collected payment, and reimbursed the department for the phone calls, and that's even without taking into account the administrative procedures necessary, such as sending out bills and dunning notices. Even though Usenet was a student-led skunkworks project, there was faculty support for this part of it. (I never asked the UNC CS department for a similar setup, but I also received significant faculty cooperation and funding for things like better serial ports for our Unix computer.)

Truscott and Ellis saw a significant advantage to Duke to being the central node in this star. In the words of the original announcement, "We avoid phone charges ourselves, and we get news sooner than anyone else." If a site was polled once a day, per the original scheme, it might take a couple of days for a Netnews post to be answered—but Duke would see the answer much more quickly. We did not appreciate the danger of central nodes: they effectively controlled what other sites could be on the network, and what they could see. To be sure, any other site could dial out on its own, but again, very few places had autodialers.

4 Protocol Design

Our next task was to design a protocol and file format. The transport mechanism had to be `uucp`; it was already there, and writing a new one would have been extremely time-consuming. We probably could have done it—I had enough experience with HASP multileaving[16] to know what was necessary—but it would have been pointless. The file format was a more interesting question, since it depended on (and determined) desired functionality, expected traffic, and more.

The first decision was easy: we used a simple, fixed-format, easy-to-parse set of header lines. This was partly in homage to the Unix tradition of the time, and partly in recognition of C's limited and painful character string-handling capabilities. We also decided that all messages would start with the letter A, to allow for easy migration to a newer file format as Netnews evolved. (A full description of the original file format is shown in Appendix A and in an RFC.[9])

The remainder of the first line was the article-ID: the originating host name, which could be up to eight characters, a period, and a serial number of up to five digits. A number of factors went into this decision. First, having a unique ID for each article made it easy to detect and drop duplicates. Our original topology included a loop; we knew that duplication could occur. But there was a more subtle implementation issue as well. 7th Edition Unix did not implement user-level locking.[18] The normal way to implement locking was via the file system; we took this a step farther by using the article-ID as the actual file name for a received article.

This inherently prevented duplicate articles from appearing on a system.

There was another implication, though: this scheme only worked for relatively small amounts of traffic, from relatively few hosts. 7th Edition Unix limited filename componentss to 14 characters; there was no room for a fully qualified hostname (although the domain name system did not yet exist, so that question wasn't even raised). And the limit of five digits came from this 14-character limit: eight characters for the hostname, plus a period, left room for only five more digits. This was not an accidental consequence; rather, it was my mistake: I estimated that the peak eventual traffic would be 1–2 articles per day, from 50–100 sites maximum, ever. It goes without saying that this was grossly wrong even in the short term.

The second header line contained the list of newsgroups to which the article was posted. Although cross-posting later came to be seen as rude, it was an intentional feature from the very beginning, and was supported by the earliest implementations.

The path line—the list of systems through which the article had passed, culminating in the username—served two purposes. A sending machine would never transmit an article to a site whose name appeared in that list. Also, the pathname was in precisely the format used by `uucp` for multihop mail forwarding, thus permitting direct replies to the poster.

The last line of the header was the posting date; the body followed immediately thereafter, with no further delimiters.

It is a fair question why we did not use a mail-like header, with From:, Date:, etc., lines. The answer is quite simple: none of us had much experience with the ARPANET, and we did not know how it formatted its email. Even if we had known, it is unclear if we would have emulated it: transmissions over our modems were at 300bps, and the

overhead would have been considerable.

Little of this was obvious to us at the meeting. Rather, it was the result of intensive experimentation. I wrote the first version in about 150 lines of Bourne shell. It supported multiple newsgroups and cross-posting; you subscribed to a newsgroup by setting a shell environment variable to the list. Since each newsgroup had its own directory, using something like

```
export NETNEWS="*"
```

or

```
export NETNEWS="NET admin social cars tri.*"
```

made it simple for the script to do something like

```
cd $NEWSHOME
groups=`echo $NETNEWS`
```

to emit the names of any directories whose names matched the shell pattern in the NETNEWS environment variable.

To find unread articles, the script did (the equivalent of)

```
newsitems=`find $groups -type f -newer $HOME/.netnews -print`
```

This would find all articles received since the last time the user had read news. Before exiting, the script would touch $HOME/.netnews to mark it with the current time. (More on this aspect below.)

There were a few more tricks. I didn?t want to display cross-posted articles more than once, so the script did something like

```
ls -tri $newsitems | sort -n | uniq
```

to list the i-node numbers of each file and delete all but the first copy of duplicates: cross-posted articles appeared as single files linked to from multiple directories. Apart from enabling this simple technique for finding duplicates, it saved disk space, at a time when disk space was expensive.

There were three major limitations to this scheme. The first was performance: on the hardware of the time, a shell script of this length and complexity was much too slow. I solved that by rewriting my script in C, though without changing any of the semantics. (Using the shell permitted what is now called rapid prototyping—on that hardware, doing even a small compilation took a significant amount of time.

The second was that only articles posted to the special newsgroup NET was sent to remote sites. That is, there could be many local newsgroups but only one remote one. That was eventually changed in the production version written by Truscott and Steve Daniel, also a Duke grad student. His version transmitted anything matching NET.* to remote sites, sending each peer site only the newsgroups to which it had subscribed.

There was a subtle problem which we realized but never fully resolved: it conflated the notions of topic and distribution. Supposed there was going to be a talk on, say, Unix system administration. That would obviously be of interest to the local system

administrators, so it should be in a system administration newsgroup. However, the distribution should be local, since no one was going to fly from California to attend such a talk.

Finally, even the production version got one thing badly wrong, echoing an error in my original design: it only tracked the high water mark of articles read. This made it impossible to read things out of order. You could skip an article to come back to it later, but the system would not remember that you had read any later articles. This wasn't a big problem as long as my traffic estimate was correct, but in that sense, Netnews was far too successful. We never planned for overwhelming success.

5 Authentication

We knew that a successful production network would require some sort of management. If nothing else, users should be able to cancel their own posts. We knew that doing this securely would require cryptography. And we deliberately did not implement anything, because we knew that we did not know how to *engineer* a secure, usable solution.

5.1 Cryptographic Authentication

Truscott, Ellis, and I knew about public key cryptography; we'd all seen Martin Gardner's column in *Scientific American*[5] and we'd all seen the original RSA paper.[19] For that matter, 7th Edition Unix included the `xsend` command for encrypted email using trapdoor knapsacks.[14] But we ran up against what was, to us, an unsolvable problem: how do you know which public key belongs to which site or user?

Today, there are obvious answers. Public key infrastructure is one. In fact, certificates had been invented at MIT,[13] but in a pre-search engine era we had no way of knowing about a bachelor's thesis we had no reason to suspect existed. Even if we had known, we had no way to operate a PKI, especially on an ongoing basis.

There are other ways that I now know. Every message could have been signed with an included public key; the site could have attached its own signature and key. But that would have dramatically increased the size of messages at a time of low bandwidth, and none of knew enough about cryptography at the time to even think of such a thing. The same is true of the resurrecting duckling protocol,[21] which wouldn't be invented for some 20 years. Furthermore, it relies on cryptographic hash functions, which did not exist at the time.

There is an interesting pair of legal footnotes to the cryptography discussion. As those of us working in cryptography in the 1990s learned, there were some obstacles to doing what we wanted. First, after the publication of their groundbreaking paper on public key cryptography,[4] Diffie and Hellman, along with Merkle, applied for and eventually received a US patent—4,200,770, issued April 29, 1980—on public key cryptography. Similarly, Rivest, Shamir, and Adleman received patent 4,405,829 on September 20, 1983. These patents all issued after the initial announcement of Netnews, but would have impacted later distribution and even use of the code afterwards.

US patent applications were not published then; we would have had no way of knowing that the applications were in progress.

The second issue would have been more troubling, had we gone ahead and deployed some cryptographic code: under American law, cryptography is considered a munition, and a license would have been required to export the code. Our uses might have qualified for a license—we only wanted to do authentication, not confidentiality—but the RSA code base would have been capable of either. It is quite possible that Netnews and its coders would have attracted the interest of Federal prosecutors.

5.2 Site-Spoofing

We also considered other ways to prevent site-spoofing. We could not come up with any that worked. Anyone could have created a Netnews article with some other site's name in the path—and per Section 4, that meant that the site being impersonated would never see the message; no one would forward it to them. Nor could anything be done by secure coding of the local Netnews command; all users had the ability to invoke the `uucp` remote execution command on their own. Arguably, large-scale reconfiguration of permissions, `setuid` or `setgid` commands, and the like, might have done the trick, but we knew that asking sites to make massive changes to the distribution's configurations would not have been acceptable.

In the end, we decided to do nothing. Promising false security is worse than leaving things insecure. (It's ironic that both Ellis and I ended up specializing in security for our careers.)

6 Behavioral Norms

In lieu of strong management mechanisms, Netnews had to rely on behavioral norms. This worked very well for a while. Since Netnews only ran on Unix timesharing systems, all users of Netnews had to have logins on such systems. Initially, those were largely in university computer science departments, corporate research labs, and the like. It's not that people in such environments are inherently better behaved; however, they were more susceptible to pressure from above.

All that said, even early on there was questionable behavior. By the late 1980s, someone who claimed to be an officer of a pro-pedophilia organization started posting to `misc.kids`, a newgroup about child-rearing. The strong negative reaction to him caused him to stop posting, but there was no equivalent to today's "block" function to keep him from reading. Even earlier, there was a neo-Nazi posting to assorted newsgroups.

Misbehavior did not have to be that extreme. When I joined Bell Labs, less than three years after the start of Netnews, my manager three levels up greeted me with "Hi, Steve, I've seen your flames on Netnews." The Oxford English Dictionary dates that usage of "flame" to 1981. (This was my early warning that things that happen online don't stay online.)

When we were planning Netnews, we were aware of the possibility of misbehavior. It was even mentioned in the original public announcement:

4. What about abuse of the network?

   In general, it will be straightforward to detect when abuse has occurred and who did it. The `uucp` system, like UNIX, is not designed to prevent abuses of overconsumption. Experience will show what uses of the net are in fact abuses, and what should be done about them.

   Certain abuses of the net can be serious indeed. As with ordinary abuses, they can be thought about, looked for, and even programmed against, but only experience will show what matters. `Uucp` provides some measure of protection. It runs as an ordinary user, and has strict access controls. It is safe to say that it poses no greater threat than that inherent in a call-in line.

5. Who would be responsible when something bad happens?

   Not us! And we do not intend that any innocent bystander be held liable either. We are looking into this matter. Suggestions are solicited.

We were worried about other abuses as well. The announcement mentions overconsumption of resources as a risk; we knew of that from an article we had seen by Dennis Ritchie.[17] Quoting him:

> The weakest area is in protecting against crashing, or at least crippling, the operation of the system. Most versions lack checks for overconsumption of certain resources, such as file space, total number of files, and number of processes (which are limited on a per-user basis in more recent versions). Running out of these things does not cause a crash, but will make the system unusable for a period. When resource exhaustion occurs, it is generally evident what happened and who was responsible, so malicious actions are detectable, but the real problem is the accidental program bug.

Note the similarity between our "it will be straightforward..." and Ritchie's conclusion.

The real message from the announcement, though, was simpler: "here is something that has properties that are a priori unknown and unknowable. Let's see what the real problems are and then figure out how to fix them." This attitude worked remarkably well for 20+ years.

7 The Public Announcement

Netnews was announced to the public in at a Usenix meeting in Boulder, Colorado, in January 1980. (The original announcement is reproduced as Appendix A.) The code base we had was the rewrite of my C version. Among the notable changes: there were multiple top-level hierarchies, not just NET.*. It was also much more designed for operation: what the remote sites were, and what newsgroups they would receive, were now stored in a file rather than being hard-coded into the program.

We also had to contend with an issue that we had not thought of originally: the unwillingness of many sites to modify the `uucp` distribution. There were, for fairly obvious security reasons, restrictions on what commands a remote site was allowed to

execute. We wanted to add a new command, rnews, to the authorized list, but that list was compiled into the `uucp` code and could not be changed by a site without changing the source and recompiling. Many sites were not willing to do that. Accordingly, we implemented a variant scheme: send email to a given email address at the remote site; a clock-driven daemon would periodically "read" the emailed news articles and post them locally.

The other striking thing is what we envisioned as the purpose of Netnews: "The first articles will probably concern bug fixes, trouble reports, and general cries for help." We also discussed locating and arranging for the distribution of software packages, though we suggested that these not be flooded to the network. To pick an arbitrary point of comparison, the source to `uucp` was about 120KB. Per the effective transfer speed estimate in the announcement, about 1000 bytes per minute, it would take two hours and cost about US$20 to send that much data. Adjusting for inflation, that's over US$60 in today's money—and most people don't want most packages but would have to receive them anyway if they were simply posted. And there was another issue: Duke only had two autodialers; there simply wasn't the bandwidth to send big files to many places, and trying to do so would block all news transfers to other sites. Instead, the proposal was for someone—Duke?—to be a central repository; software could then be retrieved on demand. That model was later adopted by UUNET.

There was one major omission in the announcement: there was no mention of the social uses of the network. We simply did not anticipate the many things that people would want to talk about with random strangers. This short-sightedness—and it was short-sighted; we certainly knew of ham radio—contributed significantly to my gross underestimate of traffic volume.

8 Network Growth and B-News

Netnews grew slowly at first. There was a chicken-and-egg problem: without more content, there was nothing to attract users, but without more users, there was no one to generate content. The ARPANET came to the rescue.

Among other things, the ARPANET had two popular, very active mailing lists, SF-LOVERS for science fiction fans and HUMAN-NETS, covering how computers interacted with society. Mary Ann Horton,[10] then a PhD student at Berkeley, set up a gateway between those mailing lists and Usenet.[†] That created the necessary traffic. It wasn't long before a protocol to transfer Netnews articles over the early Internet was developed.[12]

The increase in traffic, though, underscored the weakness of the very old decision to disallow out-of-order reading. Furthermore, the limits of the A-news file format were becoming critical; there was no easy way to add control messages for things like newsgroup creation or message cancellation. Horton and a high school student, Matt Glickman, implemented what became known as B-news; it rapidly (and rightly) displaced the original code. That story has often been told before.[7]

[†]There is some confusion over exactly who created the original gateway. I am following the history given in.[7]

A lot of the topology remained star-centric: the problem of long distance telephone calls was an expensive one. However, the central nodes (and there were generally more than one) did vary over time. At one point, Bell Labs Research was a key node, until traffic volume grew too great. Its role was taken over by the group at Digital Equipment Corporation responsible for liaison with the Unix community. Other key nodes included a Bell Labs site outside Chicago, a US government site whose administrator later set up UUNET, a separate company to provide that and other services, and more. Indeed, the closest that Usenet ever had to a governing body was the so-called "Backbone Cabal": the administrators of the key nodes plus a very few others (such as me) who were members for historical reasons. All that said, the underlying technology remained decentralized; if your local star node did not carry a newsgroup that you wanted to receive, you could always arrange to receive just that group from somewhere else.

But with that growth came controversy and the breakdown of norms. For better and for worse, growth has always been the driver in Netnews' evolution.

9 Conclusion

In looking back at the events of more than 40 years ago, it's important to place things in context. Some things just were not knowable then, even if they're obvious today.

The choice to use `uucp` over dial-up modems and a flooding algorithm were clearly correct. Other than dial-up, there were no other link technologies broadly available. Easy availability of a TCP/IP stack, especially for machines with such limited address space, was years in the future, and we would have needed to invent something like SLIP.[20] `Uucp` was the only rational choice.

A star network was also inevitable—as noted, very few places had autodialers. Even when that problem was solved, with the advent of the Hayes autodial modem, the cost of long distance calls (especially as traffic volumes grew) meant that someone with enough resources had to foot the bill. The "each node pays for their own traffic" model could have worked, but the administrative overhead of doing the billing would, I think, have been considerable and we didn't think about it enough.

We did anticipate the creation of some sort of I-have/I-want protocol, though we didn't design one, but with only daily connectivity, the latency would have been too great. Furthermore, with most of the topology being star-like, there would have been few loops: if a star node had an article and your site was not in the path, you almost certainly did not already have it.

The decision to omit cryptographic authentication was likely correct. As noted, we simply did not know enough, nor were there readily available sources of information—the major cryptography and security conferences were also in the future. Furthermore—and this is mentioned explicitly in the announcement—we were amateurs and we knew it. The legal issues would have been devastating, but since we didn't know of them we obviously did not take them into account.

Abusive behavior is a more interesting question. We realized that that could happen; we simply had no idea what forms it could take. Even in retrospect, I don't know of any papers documenting such on the ARPANET of that era. That's why we said "only experience will show what matters." For some things, there is no substitute for

the actual experiment. We expected to change things later, but wanted to learn what was wrong first: "Yes, there are problems. . . Once the net is in place, we can start a committee. And they will actually use the net, so they will know what the real problems are."

One problem that did show up as the net grew was governance. It has never been solved satisfactorily. Netnews is, by intent, distributed and decentralized, and anyone could join if they had just a single node willing to peer with them. Even if it was possible to map the net (and there were attempts to automate that, a few years later[8]), who should vote? Every user? Every system administrator of a node? Do you weight sites by the number of users at them? By the traffic they generate? By the traffic they relay? That latter would privilege the star nodes, and in fact did happen in the form of the Backbone Cabal—but that was dissolved because of complaints that it lacked legitimacy.

Our biggest failure, though, was that we never planned for success. Granted, it would have been impossible to even contemplate today's traffic volumes, given the link speeds and disk capacities of the time—as of February 2023, Newsdemon, a major commercial Netnews site, reported an average daily volume of 196 TiB.[‡] Note that the disk drives of the day had capacities measured in 10s of megabytes; even contemplating a gigabyte of storage was insane, never mind the link speeds necessary.

But we didn't even adequately plan for the traffic of a very few years later, especially in the ways we handled newsgroups (a deeper hierarchy would have been useful) or in the lack of any real capability to read things out of order. At 1–2 articles a day, that worked, but even a single order of magnitude increase made that untenable.

Did we do a good job? Netnews is still around, more than 40 years later, though it's used for very different purposes—a lot of today's traffic volume is apparently music and videos, often pirated. (MP3 and MP4 didn't exist back then, either. Indeed, the first consumer CD players weren't yet available.) It is interesting to compare Netnews to CSnet,[2,3] which is almost as old. We joked that CSnet was created by professors, who wrote proposals, got grants, etc. We were grad students—we just did it. CSnet was good for email and had some ability to retrieve files, but it did not provide for open discussion fora—and when widespread Internet connectivity became available, CSnet went away.

Today's discussion sites are either very distributed—some site will host a chat room for a particular subject—or are run by large, corporate entities such as Facebook and Twitter. Some of those, e.g., Reddit, do have multiple topic areas, but even with corporate oversight the governance problem remains difficult. And corporate governance policies can change with management. Elon Musk's takeover of Twitter is one example, but there have been radical changes of policy by some sites[1] even without new ownership. The Fediverse comes close to the original spirit of Netnews, in that it's fully distributed, but there have been complaints about the lack of overall policies on abusive content.

And thus we see the two biggest problems, ones that are not solved even today: coping with large amounts of information, and governing who can say what, where, without unduly infringing on free speech. Netnews encountered these problems first,

[‡]Statistics are from `https://www.newsdemon.com/usenet-newsgroup-feed-size`.

because it was a pioneering network, but we now know just how hard those problems are.

Acknowledgments

Tom Truscott, one of the other creators of Netnews, made many useful suggestions.

References

[1] Jon Brodkin. "YouTube now allows videos that falsely claim Trump won 2020 election". In: *Ars Technica* (June 2, 2023). URL: https://arstechnica.com/tech-policy/2023/06/youtube-now-allows-videos-that-falsely-claim-trump-won-2020-election/.

[2] Douglas Comer. "The Computer Science Research Network CSNET: A History and Status Report". In: *Commun. ACM* 26.10 (Oct. 1983), pp. 747–753. ISSN: 0001-0782. DOI: 10.1145/358413.358423. URL: https://doi.org/10.1145/358413.358423.

[3] Peter J. Denning, Anthony Hearn, and C. William Kern. "History and Overview of CSNET". In: *ACM SIGCOMM Computer Communication Review* 13.2 (Apr. 1983), pp. 138–145. URL: https://dl.acm.org/doi/abs/10.1145/1024840.1035267.

[4] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* IT-22.6 (Nov. 1976), pp. 644–654. URL: https://ieeexplore.ieee.org/abstract/document/1055638.

[5] Martin Gardner. "Mathematical Games: A new kind of cipher that would take millions of years to break". In: *Scientific American* (Aug. 1977). URL: https://www.scientificamerican.com/article/mathematical-games-1977-08/.

[6] Katie Hafner. "James T. Ellis, 45, a Developer Of Internet Discussion Network". In: *New York Times* (July 1, 2001). URL: https://www.nytimes.com/2001/07/01/us/james-t-ellis-45-a-developer-of-internet-discussion-network.html.

[7] Michael Hauben and Ronda Hauben. "On the Early Days of Usenet: The Roots of the Cooperative Online Culture. (Chapter 10)". In: *First Monday* 3.8 (Aug. 3, 1998). DOI: 10.5210/fm.v3i8.613. URL: https://firstmonday.org/ojs/index.php/fm/article/view/613.

[8] Peter Honeyman and Steven M. Bellovin. "PATHALIAS or The Care and Feeding of Relative Addresses". In: *Proc. Summer Usenix Conference*. 1986. URL: https://www.cs.columbia.edu/~smb/papers/pathalias.paper.pdf.

[9] M.R. Horton. *Standard for interchange of USENET messages*. RFC 850. IETF, June 1983. URL: http://www.rfc-editor.org/rfc/rfc850.txt.

[10] Mary Ann Horton. *Trailblazer: Lighting the Path for Transgender Equality in Corporate America*. Poway, California: Red Ace Press, 2022.

[11] A.H. Inglis and W.L. Tuffnell. "An improved telephone set". In: *Bell System Technical Journal* 30.2 (Apr. 1951), pp. 239–270. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1951.tb03659.x.

[12] B. Kantor and P. Lapsley. *Network News Transfer Protocol*. RFC 977. IETF, Feb. 1986. URL: http://www.rfc-editor.org/rfc/rfc977.txt.

[13] Loren M. Kohnfelder. "Toward a Practical Public-Key Cryptosystem". MA thesis. Department of Electrical Engineering, Massachusetts Institute of Technology, May 1978.

[14] R. Merkle and M. Hellman. "Hiding information and signatures in trapdoor knapsacks". In: *IEEE Transactions on Information Theory* 24.5 (1978), pp. 525–530. DOI: 10.1109/TIT.1978.1055927.

[15] Robert M. Metcalfe and David R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks". In: *Commun. ACM* 19.7 (July 1976), pp. 395–404. ISSN: 0001-0782. DOI: 10.1145/360248.360253. URL: https://doi.org/10.1145/360248.360253.

[16] M.R.A. Oakley and P. Hazel. *HASP "IBM 1130" multileaving remote job entry protocol with extensions as used on the University of Cambridge IBM 370/165*. Tech. rep. UCAM-CL-TR-12. University of Cambridge, Computer Laboratory, Sept. 1979. DOI: 10.48456/tr-12. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-12.pdf.

[17] Dennis M. Ritchie. "UNIX Time-Sharing System: A Retrospective". In: *Bell System Technical Journal* 57.6 (July–Aug. 1978), pp. 1947–1969.

[18] Dennis M. Ritchie and Ken Thompson. "The UNIX Time-Sharing System". In: *Commun. ACM* 17.7 (July 1974), pp. 365–375. ISSN: 0001-0782. DOI: 10.1145/361011.361061. URL: http://doi.acm.org/10.1145/361011.361061.

[19] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. "A Method of Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. URL: http://doi.acm.org/10.1145/359340.359342.

[20] J.L. Romkey. *Nonstandard for transmission of IP datagrams over serial lines: SLIP*. RFC 1055. IETF, June 1988. URL: http://www.rfc-editor.org/rfc/rfc1055.txt.

[21] Frank Stajano and Ross Anderson. "The resurrecting duckling: Security issues for ad-hoc wireless networks". In: *International workshop on security protocols*. Springer. Apr. 1999, pp. 172–182. URL: https://www.cl.cam.ac.uk/~fms27/duckling/duckling.html.

[22] Tom Truscott. “Invitation to a General Access UNIX Network”. In: *Australian Unix Users Group Newsletter*. Vol. II. IV. Australian Unix Users Group, Apr.–May 1980, pp. 15–18. URL: https://www.tuhs.org/Archive/Documentation/AUUGN/AUUGN-V02.4.pdf.

[23] Leland H. Williams. “A Functioning Computer Network for Higher Education in North Carolina”. In: *Proceedings of the December 5-7, 1972, Fall Joint Computer Conference, Part II*. AFIPS ’72 (Fall, part II). Anaheim, California: Association for Computing Machinery, Dec. 5, 1972, pp. 899–904. ISBN: 9781450379137. DOI: 10.1145/1480083.1480116. URL: https://doi.org/10.1145/1480083.1480116.

A The Original Announcement

Below is a reproduction of the original public announcement of Netnews first present at Usenix in January, 1980. This copy was taken from the April-May 1980 edition of the Australian Unix Users Group Newsletter.[22]

Invitation to a General Access UNIX* Network

Tom Truscott, Duke University

Invitation

A group of UNIX systems at Duke University and the University of North Carolina, Chapel Hill, have established a uucp-based computer communication network. Admission to the net is open to all UNIX licensees. In addition to providing the "uu" services available in the Seventh Edition of UNIX (remote mail, file transfer, job execution), it will provide a network news service. A prospective node must have a call-in facility, call-out facility, or some other means of communication with another UNIX net system. The node must have, or be able to legitimately obtain, uucp and related software.

Systems which do not call-out to the net must be polled occasionally. We will poll any system that so requests, and will bill the polled system for phone costs. The phone costs are expected to be $10-20/month. Requests for an application should be sent to

James Ellis
Department of Computer Science
Duke University
Durham, NC 27706
Telephone: (919) 684-3048

Services

The initially most significant service will be to provide a rapid access newsletter. Any node can submit an article, which will in due course propagate to all nodes. A "news" program has been designed which can perform this service. The first articles will probably concern bug fixes, trouble reports, and general cries for help. Certain categories of news, such as "have/want" articles, may become sufficiently popular as to warrant separate newsgroups. (The news program mentioned above supports newsgroups.)

The mail command provides a convenient means for responding to intriguing articles. In general, small groups of users with common interests will use mail to communicate. If the group size grows sufficiently, they would probably start an additional news group.

*UNIX is a Trademark of Bell Laboratories.

Complete programs and other machine readable text are inappropriate for transmission via (ordinary) news. On the other hand, if a news contributor announces a new version of "x.c", he could be innundated with requests such as "please mail dod!mum a copy of x.c." To avoid that, he could make his copy of x.c directly accessible to anyone via the uucp file copy program. In order to reduce uucp traffic, copies could be made at the more central nodes of the net. Traffic will be reduced further by extending news to support "news on demand." X.c would be submitted to a newsgroup (e.g. "netbulk") to which no one subscribes. Any node could then request the article by name, which would generate a sequence of news requests along the path from the requester to the contributing system. Hopefully, only a few requests would locate a copy of x.c. "News on demand" will require a network routing map at each node, but that is desirable anyway.

It is hoped that USENIX will take an active (indeed central) role in the network. There is the problem of members not on the net, so hardware newsletters should remain the standard communication method. However, use of the net for preparation of newsletters seems like a good idea.

Implementation

The hardware and software requirements for a system to join the net were mentioned above. The uucp system has been retrofitted to run on the Sixth Edition of UNIX, so both Sixth and Seventh Edition Systems can join. Each node must have a unique name, so all names must be cleared by the network administration. Duke will provide the initial administrative functions, and will also provide software for the network news to function.

Although we can supply a news program, individual sites may prefer to adapt their existing programs. This requires the ability to print and receive articles in the news transfer format. The format used to transmit an article between systems is given below. The transferred file consists of a sequence of 0 or more formatted articles, followed by a line consisting of the character `.'. The first character of the article identifies the format and will be used to simplify inevitable changes in article formats. The rest of the first line is a unique system-wide name, which also identifies the originating node. The article name is used to prevent the unlimited duplication of news articles that might otherwise occur. It has a side benefit of simplifying the implementation of a "news on demand" facility. Support for newsgroups (line 2) is required. All network newsgroups will have the prefix "net". Support for contributor name (line 3) and contribution date (line 4) are recommended. A `deletion date' is not supported; it is up to each node to delete ancient news. There is no article title per se, although the first line of the article text could be used for that purpose.

News Article Format

line	example	description
1	Aunc.173	Articles begin with the character `A´. The originating system is `unc´, which gave the article the unique name `unc.173´.
2	netsys7:netnit	Newsgroups to which this article belongs.
3	duke!unc!smb	The net path to the contributor. This line changes as the article passes from system to system.
4	Thu Jan 24 01:39:20 EST 1980	The date the article was submitted to news at the originating system.
5	Delete line 221 of cron.c:	The text of the news article.
6	*cp++ = ´\n´;	
7	Not needed. It confuses ps.	
8	..	Article terminator.

Article and newsgroup names are restricted to 14 or fewer characters. A news transfer file consists of a sequence of formatted news articles followed by `.´ alone on a line.

Questions Answered

1. Won't this be expensive?
Not at all. Night time phone costs are perhaps $0.50/3 minutes, in which time uucp could transfer perhaps 3000 bytes of data (300 baud). Daily polling would then cost $15.00/Month, which is half what Duke pays just for an office phone.

2. Could Duke really handle all the phone calls?
Sure. We have two call-out lines: at five minutes/call, we can handle 24 calls/hour. Other nodes can also volunteer to perform the call-out function.

3. What does Duke get out of this?
We avoid phone charges ourselves, and we get news sooner than anyone else.

4. What about abuse of the network?
In general, it will be straightforward to detect when abuse has occurred and who did it. The uucp system, like UNIX, is not designed to prevent abuses of overconsumption. Experience will show what uses of the net are in fact abuses, and what should be done about them.

Certain abuses of the net can be serious indeed. As with ordinary abuses, they can be thought about, looked for, and even programmed against, but only experience will show what matters. Uucp provides some measure of protection. It runs as an ordinary user, and has strict access controls. It is safe to say that it poses no greater threat than that inherent in a call-in line.

5. Who would be responsible when something bad happens?
Not us! And we do not intend that any innocent bystander be held liable either. We are looking into this matter. Suggestions are solicited.

6. This is a sloppy proposal. Let's start a committee.
No thanks! Yes, there are problems. Several amateurs collaborated on this plan. But let's get started now. Once the net is in place, we can start a committee. And they will actually use the net, so they will know what the real problems are.

7. Okay, so a few systems get the net started. What next?