ELECTRONIC PUBLISHING, VOL . 1(1), 55–76 (APRIL 1988)

The USENET cookbook—an experiment in electronic publishing

BRIAN K. REID

DEC Western Research Laboratory
Palo Alto
California
USA

SUMMARY

Much of the research taking place in the field called 'electronic publishing' would perhaps be better called 'electronic printing' or 'electronic typography' or 'electronic drawing' or 'electronic file cabinets'. The word 'publishing' has traditionally meant 'to make generally known' or 'to disseminate'. In December 1985 I began a venture in true electronic publishing—'true' in the sense that its primary goals were to explore electronic dissemination rather than electronic typesetting or formatting. I wanted to start a periodical that could be distributed electronically, that would use computers for every aspect of its production and distribution process, and that would be on a topic of wide enough interest to attract subscribers in as many countries as possible. Furthermore the topic had to be absorbing enough to engage my own interest for long enough to gain substantial experience.

The chosen topic was cookery. I began a weekly magazine whose contents are recipes. To submit a recipe for publication, a prospective author mails the recipe to the editor by electronic mail. The publishing process from that point is similar to more ordinary magazines. A copy editor rewrites the recipe for stylistic consistency and then hands it to the 'international desk', which checks to make sure that the recipe uses only ingredients that are widely known and internationally available. The international desk also converts recipes to or from metric units, so that every recipe will include both. From the international desk, the recipe goes to a 'test and proofreading' office, at which an editor checks to make sure that the recipe is coherent and comprehensible and that the dish it describes is palatable. Finally, recipes are moved to the production office, where they are bundled into issues in time to meet a Thursday publication deadline. During this test period I have done all of the editorial tasks myself, but the internal structure of the publication system is such that different people could do the different tasks without disrupting the flows and procedures.

The recipes are distributed in a text formatting language, and each subscriber is sent software to format that language into some output format that he can print on his machine. Subscribers typically extract the recipes out of each weekly issue and put them into a local database, from which they can print pages for a notebook or access the recipes with online retrieval commands. The text formatting language is a dialect of *troff*, and the vast majority of subscribers use a special set of *troff* macros to do the formatting. The publication is called *The USENET Cookbook*. It has about 13000 subscribers worldwide, and has had recipes contributed by about 300 different people. Most of the subscribers are in English-speaking countries.

KEY WORDS Magazine Automated production Online publication Cookbook

0894–3982/88/010055–22$11.00

Received 8 December 1987
Revised 22 December 1987

INTRODUCTION

Most modern communication networks occupy the majority of their bandwidth with discussions about the network itself or the technology behind that network. The air time of amateur radio is almost entirely devoted to discussions of amateur radio equipment (and indeed in many countries it is illegal to discuss much else). The bandwidth of most global message systems is devoted in large part to discussions of the message systems themselves, especially their reliability. USENET is probably the largest global computer-based broadcast network; according to periodic measurements about 30% of the capacity of USENET is used for discussing the network itself, and another 30% is devoted to discussions of the computers out of which USENET is built[1,2].

In the beginning, global networks required such intense technological expertise that only technologists could participate, and as every technologist's spouse has learned at office parties, most technologists are primarily interested in discussing technology. USENET has reached a level of simplicity at which it can be used profitably by people who are not professional technologists, though it certainly cannot be used by people who are uncomfortable and untrained with computers.

In 1985 I decided to begin an experiment in 'true' electronic publishing, in keeping with the dictionary definition of the word 'publish'. Specifically, I wanted to ignore issues of typesetting, graphics, text formatting, and fonts, and to explore the use of a global computer network to distribute a magazine devoted to a topic having nothing to do with computers or technology. The subject of food and cooking is ideal, because graphics and high-technology typesetting are not mandatory for discussing food, because most computer scientists and many USENET users are interested in food, and also because I was interested in food: I would be the magazine's editor-in-chief, and I needed to pick a topic in which I could sustain interest for several years.

I created a weekly publication called *The USENET Cookbook*, which is published entirely by electronic distribution. It is devoted to the distribution of recipes and food lore, and consists primarily of recipes that were submitted by readers, tested and rewritten by me, and bundled together into weekly issues.

The USENET Cookbook is not a commercial venture, but it is venturing into a domain (publication) that is normally the province of commercial concerns. Like most projects mounted in a research laboratory, *The USENET Cookbook* is an experiment. It is important that the reader understand the nature of the experiment, or perhaps more importantly, its scope. *The USENET Cookbook* is not an experiment in the commercial viability of anything. The standard technique for testing commercial viability is to start a new company attempting to engage in that business, and see if it makes or loses money.

The primary focus of the experiment is to create something that uses a lot of new technology but otherwise resembles a conventional publication, to run it for a while, and to see what happens. I was not entirely sure what I wanted to learn. I feel sure that Dr. Frankenstein had very similar goals for constructing his monster: make something that is a reasonable laboratory replica of a well-known object, let it loose for a while, and see what happens. Naturally I hope that the results of letting *The USENET Cookbook* loose for a while are less traumatic.

Because the experiment is so tentative, it is not reasonable to charge money. The operative goal was to get a lot of readers and to find out whether or not they think the

experimental publication is offering them something of value. If the result turns out to be something valuable, then perhaps, someday, some form of 'technology transfer' can be done to move from laboratory-curiosity magazines running on experimental networks into professional-quality magazines running on common-carrier delivery systems. It would be mad to attempt a commercial publication venture without having first dabbled in scaled-down laboratory versions.

An obvious question is whether or not anything learned from this experiment could be applied in a commercial setting. Naturally it depends on the nature and technology of the commercial setting. The success of satellite television programs that are encrypted for broadcast, and decryptable only by paying subscribers, is proof that an encrypted pay-for-service broadcast medium can work.[1] I consider it important to separate the testing of the technology from any test of consumer interest in the technology. *The USENET Cookbook* is testing technology.

As I write this introduction, the 100th weekly issue of *The USENET Cookbook* is poised ready to go out. In the two years that *The USENET Cookbook* has been in operation, I have learned a lot about how to engineer, build, and run such a venture. This paper describes the architecture of the publication system, my experience with running it, and my conclusions after running it for two years.

SYSTEM STRUCTURE

The USENET Cookbook is a distributed system, and its overall makeup is reasonably complex. Therefore I shall describe its structure first from the reader's point of view, and then from the publisher's point of view. The full details are given in the section on Publication Mechanisms on page 64.

Structure from reader's viewpoint

In order to subscribe to *The USENET Cookbook*, you must be able to use a computer that is connected to one of the international computer networks. That computer must be able to receive USENET, or else it must be able to receive electronic mail from the global internetwork (e.g. ARPAnet, Internet, uucp, CSnet, ACSnet, JANET, etc.)[1]. Each subscriber to *The USENET Cookbook* sees the publication as a weekly periodical, which arrives on Friday in North America and on the weekend elsewhere. Once each week an issue arrives, with each issue consisting of an introductory message followed by a series of electronic mail messages and USENET articles.

The subscriber has several options, depending on his level of technological sophistication and on his interest. He can thumb through the messages manually, inspecting them and keeping the ones he likes. He can arrange to have all incoming recipes automatically kept in a database directory. He can also arrange, on a timesharing machine or network file system, to have all incoming recipes stored in a public database so that others can use them.

Each article arrives in two different formats, first in a plain text version and then in a version encoded in a text-formatting language. If the subscriber's computer system is not

[1] Most North American satellite TV channels are encrypted using a scheme called VideoCypher II. The ability of a customer's television to receive a program can be switched on or off from the broadcast transmitter.

able, for one reason or another, to process the encoded versions, then the he can keep the plain-text versions of articles that he likes. If he has the software for formatting the encoded versions, then he keeps those versions, and can produce plain-text, typeset, laser printed, or other versions from the encoded form.

Subscribers whose computing technology is primitive, e.g. those participating via simple personal computers, usually must be satisfied just with collecting the plain-text recipes. If the subscriber can run Unix and can install a 'subscriber software package' on his machine, then he can make indexes, cross-references, online 'man page' versions, and so forth.

It is important to note that although the subscriber can use Unix programs to make better use of the recipes, you do not have to have a Unix computer (or know how to use one) to be a subscriber. The local Unix programmability is an extra-value option. Many subscribers are programmers or computer scientists, of course, and they are quite capable of writing their own software locally to process the recipe and article database as they wish. A surprising fraction of the subscribers are not at all technologically sophisticated; many of them are not even capable of installing the canned software package which is distributed to subscribers. Many more subscribers, of course, use some non-Unix computer to read and store the recipes, and the system must be able to work for them as well.

The text-formatting-language versions of the articles have both U.S.-style measurements (cups and spoons) and pure metric measurements (grams and milliliters) in their text. The formatting programs select one set of measurements or the other, at the subscriber's request. The plain-text versions of the articles all use U.S.-style measurements.

Structure from publisher's viewpoint

A later section gives technical details of the automated publishing mechanism, but here is a brief overview for readers who are not already familiar with *The USENET Cookbook*. Like almost all periodicals, *The USENET Cookbook* is centrally published. This means that there is a central editorial office through which all articles and editorial material are filtered, and that the actual publication process begins by shipping one copy of the final form of the periodical to a distribution network, which makes copies as appropriate and then distributes those copies.

Readers of *The USENET Cookbook* submit articles to it by mailing the articles to the editor. They arrive by electronic mail. The editor performs an initial screening of each submission, then hands it to the editorial department. The editorial department has a series of stations, each of which has an 'in' queue and an 'out' box. Each submitted article moves through the editorial process, station by station, until it is ready for publication. At that point it is put into a 'ready to publish' queue.

Once a week, on Thursday evenings, an automatic batch job assembles an issue by removing an appropriate number of articles from the 'ready to publish' queue, forming them into a package, and transmitting them over various computer networks as appropriate.

At the moment the entire editorial staff consists of one person, who does all of the work at all of the stations, but this is a closely guarded secret. Because the processing is carefully divided into steps, each of which has a separate input queue and output port, the

editorial work can be split among several people if the magnitude of the project ever warrants it.

Network mechanisms

There are three different distribution mechanisms used to publish *The USENET Cookbook*: USENET newsgroups, direct electronic mail, and on-demand electronic mail. Each of these is a relatively ordinary mechanism and does not warrant detailed discussion here, but for those readers not familiar with them, a brief explanation is in order.

USENET is a worldwide bulletin board system built on top of many different kinds of actual network electronics. It 'broadcasts' every article to every node of USENET by the simple device of having every node send every article to all of its neighbors who might not have seen it yet. Some crude heuristics permit this 'flood fill' broadcasting mechanism to work without a lot of redundant retransmission. USENET was originally developed in 1980 by a few people at the Research Triangle Institute in North Carolina, and has grown by about 30% per year ever since.

USENET articles are categorized and stored in 'newsgroups'; a newsgroup corresponds roughly to a file system directory or to a fixed keyword in a keyword-based bulletin board. There are approximately 300 different USENET newsgroups. Every newsgroup is either 'moderated' or 'unmoderated'. To post an article to a moderated group, an author must mail the article to the newsgroup's moderator, who will either post it or reject it. To post an article to an unmoderated group, an author simply puts the article into the news database on his local machine, and the flood-fill software takes care of further distribution. There are about 8000 sites on USENET worldwide, with about 250 000 people actively reading one or more newsgroup[2].

Direct electronic mail is a worldwide monolith cobbled together from a number of independently-administered networks. Although every network has its own set of protocols, it is almost universally possible to build mail-transfer gateways between networks, and in general those gateways exist. Therefore it is usually possible, but rarely easy, to send electronic mail from the publishing center of *The USENET Cookbook* to any network-connected machine. When each week's issue goes out, the majority of the subscribers receive it by the vastly less expensive flood-fill USENET conduit. There is a small mailing list of people who receive *The USENET Cookbook* by electronic mail. Because electronic mail subscriptions are so much more expensive to maintain than broadcast subscriptions, the mailing list is intentionally difficult to join, and if two successive mailings to any recipient are returned as undeliverable, that recipient is automatically removed from the mailing list.

The third distribution channel, on-demand electronic mail, is primarily for the purpose of handling 'back issue' requests. However, several hundred people use it as their primary means of subscription. It is described more fully in the later section on the Archive Server.

PUBLICATION PROCEDURE

This section describes the procedures that are to be followed to process articles submitted for publication and to prepare weekly and special issues. The mechanisms that implement these procedures are described in the next section.

```
From: mcvl@src.dec.com (Mary-Claire van Leunen)
Date:  6 Jul 1987 1519-PDT (Monday)
To: recipes@decwrl.dec.com
Cc: mcvl
Subject: roasted red peppers

Safeway got in their first carload of really cheap red peppers yesterday,
and I made roasted pepper salad:  Turn red peppers under the broiler till
their skins come up in black bubbles.   Cool the peppers till you can handle
them comfortably.   Remove the skins, stems, cores, and seeds.  (Messy work.
I have tried removing the innards first, before putting the peppers under
the broiler, and the result is not so good as this way.)
Cut the peppers into finger shapes.   Flavor them with fresh lemon juice and
salt.   Let them sit for several hours, draining off the liquid that forms.
Serve them at room temperature as a side dish for anything Nicoise or
Spanish or Greek or Egyptian.  Or make a meal of them by adding a bowl
of yoghurt, a bowl of chopped onion, some black olives, radishes, and
pistachios, and lots of warm pita bread.   Or use them with sauteed garlic
and pitted black olives and a little reduced white wine as a spaghetti sauce.
Now.   Here is the real point about this recipe:  These roasted peppers
are insanely delicious, smooth and unctuous and aromatic, and yet
they contain absolutely no calories.   A full-sized red pepper is only
15 calories.   How can this be?
```

Figure 1. A narrative submission

```
Date: Wed, 2 Dec 87 14:25:39 EST
From: munnari!murdu.oz.au!graham@uunet.UU.NET (Graham Tongs)
To: recipes@decwrl.dec.com
Subject: Icecream Christmas Pudding

We here in the Southern Hemisphere have a summer Christmas so we have
adapted some of the more traditional festive fare to local conditions.
The following is one such example you may like to try - it is very easy
as it can be made up to a week ahead of using.  It has been adapted for
the Microwave but would be very simple to make without one.

FROZEN PLUM PUDDING
500 grams Mixed Dried Fruits finely chopped
1 teaspoon grated orange rind
2 tablespoons Cocoa
1/2 cup Brandy or Marsala
2 teaspoons Gelatine
1 tablespoon Cold Water
2 litres Vanilla Ice Cream

Combine fruit, spice, orange rind, cocoa and brandy in a large microwave dish.
Cover and cook on HIGH 3 mins, stirring once.   Cool, stirring occasionally.
Mix gelatine and water in a small bowl, cook on HIGH 30 secs.  to melt.
Blend into fruit, chill.  Soften Ice Cream on HIGH 1 minute, do not allow to
melt.   Fold in fruit, blending evenly. Pack into foil lined bowl, cover with
foil and freeze until required. Unmould onto a serving plate, cut into wedges
and serve with cinnamon flavoured whipped cream.  Serves 8 to 10 people.

Regards,
Graham Tongs
```

Figure 2. A structured submission

```
Date: Fri, 30 Oct 87 03:39:22 mst
From: hpcnof!kra@hplabs.HP.COM
To: recipes@decwrl.dec.com
Subject: bon-bons

.RH MOD.RECIPES-SOURCE BON-BONS D "Oct 30 86"
.RZ "PEANUT BUTTER BON BONS" "Wonderful holiday treat"
.IH
.IG "3 lb" "powdered sugar"
.IG "1 lb" "butter"
.IG "2 cup" "peanut butter, crunchy"
.IG "1 lb" "chocolate chips"
.IG "1/2" "block paraffin"
.PH
.SK 1
Melt butter, pour over sugar and peanut butter. Stir and knead until smooth.
.SK 2
Roll into walnut-sized balls.
.SK 3
Melt together chocolate chips and paraffin over water.
.SK 4
Dip balls into chocolate coating, set on waxed paper to harden.
.WR
Katherine Rives Albitz
Hewlett-Packard, Ft.  Collins, Colorado, USA
hplabs!hpfcla!hpcnof!k_albitz
```

Figure 3. A formatted submission

Submission

All articles that are published were originally submitted for publication by a prospective author. To submit an article, the author mails it by electronic mail to *The USENET Cookbook*'s editor at the Internet address `Recipes@decwrl.DEC.COM`, or at the uucp address `{decvax,ihnp4,pyramid,sun,ucbvax}!decwrl!recipes`. The nature of the submitted article varies widely. Some arrive in narrative form (see Figure 1), some arrive in tabular form (see Figure 2), and some arrive already formatted in some variant of *troff* (see Figure 3),

An editorial goal of *The USENET Cookbook* is that the recipes all have a similar style and structure, similar terminology, and consistent measurements. A significant fraction of the editorial process is devoted to achieving this goal.

When an article submission arrives, the mailer files it chronologically, logs it, and places it in an incoming mailbox queue. Two physical copies of the submission are made. This is not so much to protect against disk failure, but to remove from the editor the onus of saving a copy of the raw submission, so that the proofreader can compare the edited article against the original submission as a guard against errors being introduced in the editing process.

At intervals, an 'acquisitions editor' reads the incoming mailbox and sends an acknowledgment to the author. The acquisitions editor checks to make sure that the submission includes the author's name, address, and Internet-format electronic mail address; if not, it is the acquisitions editor's job to find them. When the acquisitions editor is satisfied that the raw information is present, he files the article in the 'submissions' queue. The copy editor processes the submissions queue.

Editing

The copy editor processes the articles that are in the submissions queue, rewrites them as needed, 'internationalizes' them, submits them for proofreading, fixes mistakes that the proofreader finds, and then hands them to the publisher by placing them in one of the 'ready to publish' queues.

The copy editor's first job is to rewrite the article to have the standard style and structure and standard vocabulary. Copy editors the world over perform a similar task, and there is very little that is different about the way that it is done for *The USENET Cookbook*. Because the standard style for *The USENET Cookbook* articles is a relatively terse, active, present-tense style, a recipe submitted in narrative form often requires a tremendous amount of rewriting. Because some articles require significantly more editing than others, the copy editor is not expected to process the submission queue in order. A tough narrative submission can sit there for months, challenging the editor to rewrite it into the house style, while formatted submissions go out in a week or two.

Once the article has been rewritten in the correct style and has been put into draft form, the copy editor then checks for problems in international units, measures, and ingredients. Every article is published with both American-style measurements (cups and spoons) and 'pure metric' measurements (grams and milliliters). The copy editor is responsible for converting to or from metric measurements, as necessary.

There is a proofreading program available to the copy editor, which he can run as frequently as he likes. The proofreading program does not check language or style, but checks spelling, punctuation, character set, syntax, and many other mechanical things. When the proofreading program passes an article, it adds an 'Approved' stamp to the text of the article, which is the signal to the copy editor that he should now pass that article to a publication queue. If the copy editor changes the article in any way after the 'Approved' stamp has been added, he is expected to re-run the proofreader.

Copy editing is done with a text editor which has been augmented with special customized commands for editing articles about food and cooking. For example, there is a command that will automatically replace 11 different variants of the word 'tablespoon' with the standard abbreviation 'Tbsp', and there is a command which turns strings like '1 pint' into '2 cups'.[2] The text editor is also integrated with a database containing all previously-published recipes, so that the copy editor can easily check them. Having rapid, single-command access to database retrieval inside the text editor is an invaluable timesaver for the copy editor. Mechanized proofreading is a tremendous boon to a non-professional weekly publication, because it remembers standards from one week to the next. It is difficult to be a stern copy editor when one only does it a few hours a week, and it is very hard to remember, from one week to the next, whether the ingredient list should say '1 garlic clove' or '1 clove garlic'. By embodying the rules in a mechanized proofreader, *The USENET Cookbook* can achieve a much greater consistency than would otherwise be possible in an amateur periodical.

Internationalization

The USENET Cookbook has been published for more than 2 years, and most of the details

[2] A pint is a varying unit of measure, but almost all of the people who don't know this live in the U.S., and therefore think it is 2 cups. An Imperial Pint is about 1.2 U.S. pints.

relating to its electronic publishing have been ironed out, but every month brings a new challenge in editing recipes so that they can be understood and used worldwide.

To 'internationalize' a recipe, the editor must attend to these details:

- *Measurements*: the recipe must have both cups/spoons and metric measurements. Since metric measurements are often by weight and not by volume, this cannot be done by blind transformation. For example, '1 cup walnuts' becomes '100 grams walnuts' rather than '250 milliliters walnuts'. Despite the 'universality' of the metric system, every country uses a different version of it. In Europe, most cooks seem to prefer 'centiliters' and 'deciliters', while in New Zealand, virtually all books say 'milliliters'. On the same day in 1986 I got a complaint from Sweden that '100 ml' should have been written '1 dl', and a complaint from New Zealand that '25 cl' did not correspond to any unit of measure that he was comfortable with. In many countries people use an eclectic mixture of grams, liters, teaspoons, and cups, but the volume of a teaspoon varies widely. The only solution is to use 'pure metric', and let each reader translate into his own country's idiom.
- *Ingredients*: most Americans have never heard of a 'digestive biscuit', and most English have never heard of a 'Graham Cracker'. The two are nearly identical. American 'Crisco' is nearly unknown outside our continent, while 'Copha', a similar substance made from (and tasting of) coconut, is virtually unknown in the northern hemisphere. There are dozens of ingredients that some countries take for granted and others have never seen. When recipes require these ingredients, they must be handled with great care.
- *Temperatures*: not just Fahrenheit and Celsius, but also 'Regulo 6' as a unit of gas heat or 'Fast oven' as a baking temperature.
- *Normal packaging*: A surprising number of recipes call for '1 box of soup mix' or '1 jar of olives'. These units must be converted into ordinary measurements, often by engaging in a long dialog with the author. The proofreading program (see next section and previous section) is able to spot almost all errors in internationalization and warn the copy editor about them. It is not able to fix them.

In working on the internationalization of recipes, the copy editor is helped tremendously by having a simple computation and database system connected to text editor commands. An editor command will automatically convert degrees Fahrenheit to degrees Celsius, or the reverse. Another converts centimeters to or from inches. Still another consults a database of foodstuff densities to convert '1/4 cup raisins' into '40 g raisins'. Whenever I learn new information about some ingredient or density or package, that information goes into the editorial database so that it will be available for use next time.

Proofreading

There is no substitute for a human proofreader, but to make the job less demanding, the copy editor can invoke a mechanized proofreading program which will flag many types of errors (though it will not fix them). The proofreading program checks that the article has the correct format, it checks spelling, it checks for the presence of correct-looking metric and U.S. measurements, it checks for the use of certain forbidden words (such as 'pint'). It checks several dozen other things. If the proofreading program approves an

article, it edits the article text to add an approval stamp; if not, it pops up a window with the error messages.

The proofreading program is in no way an artificial intelligence program or a so-called 'expert system'. It is a (relatively complex) pattern matcher and parser. Perhaps someday if artificial intelligence programs can actually be made to work as claimed, then some AI proofreader could be used. For the nonce, I shall do what most practitioners of AI do: I shall call my database system an expert system, and no one will be the wiser.

Publishing

The publishing process itself is completely automated. If the queues of publishable articles are full enough, *The USENET Cookbook* can run completely unattended for weeks. This serves two purposes. The most important and most obvious reason is that the periodical publication takes place whether or not I am available to do the work, so that it is genuinely periodical. The second and more subtle reason is that by having an automated publishing process begin of its own accord at a standard time every week, the concept of a 'press deadline' is created. Publishers have known for centuries that their staffs work harder when there are press deadlines. The relentless weekly Thursday night publication is a reminder that the editorial staff must not let the work slip for too long.

Special issues

In addition to the regularly-scheduled weekly issues of *The USENET Cookbook*, there are a few other issues. There are 4 yearly 'holiday special issues' for Passover, Easter, Thanksgiving, and Christmas, and 2 yearly 'seasonal special issues' for summer barbecues and winter soups. From time to time I invent other special issues, such as an 'april fool' special issue to hold a muskrat recipe and a recipe for margaritas made from limeade and vodka.

Note that seasons are very country-specific: subscribers in Australia and New Zealand are not able to make much use of the July barbie issue and the January soup issue, but perhaps they save the recipes until the appropriate season. Note also that November Thanksgiving is strictly an American holiday, but the principles to which it is devoted—the overconsumption of poultry and squash—apply worldwide.

Besides the issues devoted to the content of *The USENET Cookbook*, there are also some special issues devoted to propagating its mechanism. Twice a year we send out a special issue whose contents is the set of Unix programs already mentioned, and four times a year we send out a special issue containing the instructions for preparing and submitting articles.

Reader correspondence

No publication can exist without reader correspondence. The 'letters to the editor' column is a fixture of every newspaper and magazine; letters to the circulation department keep the publisher's distribution network appraised of the migrations of subscribers and help them locate back issues that they have missed or lost.

The USENET Cookbook does not have a 'letters to the editor' column. Probably it should. Since it grew up inside USENET, and USENET itself is a fine mechanism for the

publication and exchange of letters, I never felt the need to have a formal ‘letters’ department. When people have a comment about *The USENET Cookbook*, they simply publish it themselves in the form of an article to an unmoderated USENET newsgroup.

The ‘circulation department’ of *The USENET Cookbook* is handled by a completely automated mail-response server. People mail in requests for back issues of recipes, software, or documentation. The server locates their request in its database and mails it back to them, or else mails an explanation of why it was unable to find the item that they asked for. The mail-response server uses a very elaborate system of quotas and priorities to make sure that it is equitably available to every subscriber. Figure 4 shows a typical interaction with the back-issue mail response server.

```
From reid  Tue Dec  8 21:48:23 1987
Received: by woodpecker.DEC.COM (5.54.4/4.7.34)
        id AA24423; Tue, 8 Dec 87 21:48:23 PST
From: reid (Brian Reid)
Message-Id: <8712090548.AA24423@woodpecker.DEC.COM>
Date:  8 Dec 1987 2148-PST (Tuesday)
To: archive-server

send recipes fondue-1 chicken-lemon

To: reid (brian reid)
From: DECWRL archive service <archive-server>
Date: Tue, 8 Dec 87 21:52:05 PST
Subject: ack receipt of your archive retrieval request
In-Reply-To: Request from reid (brian reid) dated Tue Dec 8 21:51:08 PST 1987

The DECWRL archive server has received your request.
All of the files that you asked for are in the database.
Your request is 6987 bytes.  It has been placed in the mailer's work
queue.  The work queue is periodically processed by the archive mailer, which
always sends the shortest queued request first.  The length of time that it
takes to receive your request will depend on how many smaller requests are in
front of it in line.
```

Figure 4. A typical exchange with the archive server

A computer user can subscribe to *The USENET Cookbook* by electronic mail if he can prove to the mail-response server that mail to him (or to some other mailbox) will be deliverable. He does this by engaging it in a 4-way handshake:

1. User sends a message to the mail-response server saying ‘I would like to subscribe in the name XYZ’.
2. Mail-response server sends back to XYZ a message saying ‘OK. Here is an encrypted token. If you can prove to me that you received this token, by mailing it back to me, then I will add XYZ to the mailing list’.
3. User receives the token and mails it back to the server as proof that the mail address works.
4. The server adds XYZ to the mailing list and mails it an acknowledgment saying that it has done so.

Every mailing from *The USENET Cookbook*, whether part of a publication or in response to a specific inquiry, is annotated with a ‘subscriber code number’. If a message bounces back as undeliverable, the subscriber database is annotated to say that the mail

did not get through. If the database is already so annotated, then the subscriber is removed from the database. Undeliverable messages are the curse of global internetwork electronic mail.

A separate server command will remove a subscriber from the mailing list. There is no protection against one subscriber maliciously removing another from the list.

PUBLICATION MECHANISM

In previous sections I have described the procedure by which *The USENET Cookbook* is produced and published. In this section I describe the highlights of the technology which implements this procedure.

Recipe representation

Recipes are encoded in an extremely primitive text formatting language that is a dialect of *troff*[3]. Most Unix users have *troff*; it was originally shipped as a part of Unix (though it has recently been unbundled as part of an extra-cost package). Other text formatting languages exist, but *troff* is the only one that is widely available, able to produce output for a wide range of printing devices, and relatively simple. Because *troff* is unusable in raw form, various divergent macro packages have evolved for it, and almost no macro package is universally available. The one exception to this is that the macro package used to format the Unix manuals *is* almost universally available (though some vendors have mucked with it). Because Unix manual pages are so universal and because the macro package to implement them is so simple, I chose to represent recipes for *The USENET Cookbook* in terms of a macro package layered on top of the Unix manual macros. This has the side effect, seen by some subscribers as a strength and by others as a weakness, that the formatted recipes look alarmingly like Unix manual pages.

To facilitate mechanized postprocessing of the recipes, such as ingredient cross-referencing, they are structured in a very primitive, very rigid line-oriented style. There is a required sequence of formatting commands which every recipe must contain. Every ingredient is on a separate line, and every such line obeys rigid conventions for the use of quotation marks and special characters. As a result, it does not take much programming skill to write simple database programs to do interesting things with a database full of published recipes.

Each article, whether recipe, essay, or software distribution, is stored with a textual header which serves simultaneously as a processing checklist, an electronic mail header, a USENET distribution header, and keys to identify the article to subscriber software. This was easy because USENET distribution obeys the Internet mail envelope standard RFC822[4], and that standard permits arbitrary keyword/value fields to be in the message header. For external distribution, fields such as `To:` or `From:` or `Newsgroups:` are important; for internal processing, fields such as `Approved:` or `Replied:` or `Accepted:` are used to mark the progress of the article through the publishing process.

Figure 5 shows the entire text of an article as it sits in a publication queue waiting to be published. It has been through the complete editorial process and has been passed by the proofreading program. The lines before the copyright notice are the RFC822 header. The text of the article itself starts with the `.RH` line. This line also gives the database retrieval key (file name in which the article is stored) for this article, the creation date, and the

```
From: ed@mtxinu (Ed Gould)
Newsgroups: alt.gourmand
Subject: RECIPE: Curried peanut chicken
Date: 30 Jan 87 04:47:22 GMT
Organization: mt Xinu, Berkeley, California, USA
Approved: reid

             Copyright (C) 1987 USENET Community Trust
Permission to copy without fee all or part of this material is granted
provided that the copies are not made or distributed for direct commercial
advantage, the USENET copyright notice and the title of the newsgroup and
its date appear, and notice is given that copying is by permission of
the USENET Community Trust or the original contributor.

.RH MOD.RECIPES-SOURCE CHICKEN-PEANUT M "28 Oct 86" 1987
.RZ "CURRIED PEANUT CHICKEN" "Chicken in a spicy peanut sauce"
I got this recipe from my cousin who lives in Amsterdam.
She got it from a book of Jewish recipes from Curacao.
.IH "Serves 4"
.IG "2" "Small frying chickens,"
cut into serving-size pieces
.IG "\(14 cup" "oil" "60 ml"
.IG "2 tsp" "salt," "10 ml"
or less, to taste
.IG "3 tsp" "curry powder," "15 ml"
or more, to taste
.IG "1" "large onion,"
sliced
.IG "1" "large green pepper,"
cut into strips
.IG "1 large" "tomato,"
skinned and sliced
.IG "\(14 cup" "crunchy peanut butter" "50 g"
.IG "\(14 cup" "water" "60 ml"
.PH
.SK 1
Brown the chicken pieces in the oil.
Stir the salt and curry powder into the drippings and cook,
stirring, for one minute.
.SK 2
Add onion, pepper, and tomato.  Cover and simmer five minutes.
Add chicken pieces, cover and simmer 30 minutes or until tender.
.SK 3
Remove chicken pieces to a dish, or over rice.
Blend peanut butter with water and stir into gravy.
Heat to boiling while stirring constantly.
Serve.
.SH RATING
.I Difficulty:
easy.
.I Time:
45 minutes.
.I Precision:
no need to measure.
.WR
Ed Gould
mt Xinu, Berkeley, California, USA
{ucbvax,decvax}!mtxinu!ed
```

Figure 5. An article as represented in the database

copyright date. The publication date is in the RFC822 header. As you can see by comparing the ingredient lines for the green pepper[3] and the tomato, the proofreading program is not omniscient. It doesn't always spot the difference between '1 cup' (which is fine) and '1 large' (which is not entirely fine). The `.RZ` line is used both as a subheading in the printing of the recipe and as keyword-providing arguments to the index generation part of the subscriber software. The remaining lines are commands to the text formatter. Notice that there are no actual *troff* commands here: everything is a call to some macro or another.

Work queues

Every editorial work process has one or more input queues and appends its results to one or more output queues. Each queue is implemented as a Unix directory. There is a file named LASTIN and a file named LASTOUT. Each such file contains an integer. The articles in the queue are files with integer names. The front of the queue is the file whose name is one greater than the contents of LASTOUT, and the tail of the queue is the file whose name is one greater than the contents of LASTIN. Simple shell scripts implement the 'append', 'test if empty', and 'get next' operations. There is also an 'unget' operation, which returns an article back to the front of the queue from which it came. This allows an editor to be called to dinner in the middle of editing an article without leaving the queues in an inconsistent state. There is a semaphore for each queue, maintained as a file named `.lock`. Standard Unix file locking conventions are used.

Most work processes have three input queues and three output queues. The three queues are 'normal', 'priority', and 'background'. The priority queues are used for rushing an article through the editorial process in order to meet a specific schedule deadline; for example, a Christmas recipe submitted in early December would have to be rushed through the editorial process in order to be published by Christmas of the same year. The background queues are used for holding articles that need an unusual amount of rewriting or research, or that are being held waiting for some missing information.

Editing and proofreading

As explained in the section on Editing Procedure, the *The USENET Cookbook* editorial process consists of a series of processing stations, each of which receives submissions from a queue and sends its output to another queue.

Each processing station is implemented as a Unix directory; in that directory, the human editor uses a version of the Emacs text editor[5] which has been customized for editing this publication. The directory serves as a 'desktop', so that the copy editor can leave one job 'in progress' without returning it to a queue, simply by leaving its files in the directory. The input queues are files named 'queue', 'backgroundqueue', and 'priorityqueue' in the directory. These queue files are symbolic links to the true queue storage, but they enable a uniform input interface to the processing station. The processing stations are named 'acquisitions', 'submissions', 'rewrite', 'proof,[D,' 'publish,' and 'special,' and the directories that contain them are named accordingly.

Since all of the queues have the same internal structure, the same utility programs are used to list the contents of each. An overall program `sq` (show queues) lists the full set of

[3] A note for readers down under: a *green pepper* is a capsicum.

CHICKEN-PEANUT(M) USENET Cookbook CHICKEN-PEANUT(M)

CURRIED PEANUT CHICKEN

CHICKEN-PEANUT – Chicken in a spicy peanut sauce

I got this recipe from my cousin who lives in Amsterdam. She got it from a book of Jewish recipes from Curacao.

INGREDIENTS (Serves 4)

2	*Small frying chickens*, cut into serving-size pieces
60 ml	*oil*
10ml	*salt*, or less, to taste
15 ml	*curry powder*, or more, to taste
1	*large onion*, sliced
1	*large green pepper*, cut into strips
1 large	*tomato*, skinned and sliced
50 g	*crunchy peanut butter*
60 ml	*water*

PROCEDURE

(1) Brown the chicken pieces in the oil. Stir the salt and curry powder into the drippings and cook, stirring, for one minute.
(2) Add onion, pepper, and tomato. Cover and simmer five minutes. Add chicken pieces, cover and simmer 30 minutes or until tender.
(3) Remove chicken pieces to a dish, or over rice. Blend peanut butter with water and stir into gravy. Heat to boiling while stirring constantly. Serve.

RATING

Difficulty: easy. *Time:* 45 minutes. *Precision:* no need to measure.

CONTRIBUTOR

Ed Gould
mt Xinu, Berkeley, California, USA
{ucbvax,decvax}!mtxinu!ed

Figure 6. An article typeset by a subscriber

queues or specific subsets. Figure 7 shows a listing of the publication queues and the special-event queues. The ‘AOCM’ information is a summary of the status flags in each article; since these are publication queues shown, all of the editing operations have been performed and the corresponding status flags are set in the article headers. The Thanksgiving recipes are still shown even though Thanksgiving has passed, because the special-issue queues are set up so that some articles can be re-used, if needed, from one year to the next. The articles are not deleted from holiday and special-issue queues when they are published.

```
publish queue status at  Sun Dec 6 12:16:24 PST 1987:

in /udir/recipes/queue:

+++ To go out 10 Dec 87:
440 GRANOLA        C  AOCM patc@tekcrl.tek.c 27 Sep 87 SESAME GRANOLA
441 SPAG-SQUASH    M  AOCM horton@reed.uucp  30 Oct 87 SPAGHETTI SQUASH TETRAZ

in /udir/recipes/backgroundqueue:
  4 MUSTARD-GREENS V  AOCM nemo@rochester    18 Oct 84 MUSTARD GREENS
 27 ASPARAG-SOUP-1 SPVAOCM reid@decwrl.DEC.C  9 Mar 87 SUMMER ASPARAGUS SOUP

special queue status at  Sun Dec 6 12:16:29 PST 1987:

in /udir/recipes/special/aprilfool:
  1 GARLIC-BREAD-2 B  AOCM dkw1@hou2a        10 Jul 86 SIMPLE GARLIC BREAD
  2 MARGARITA-1    L  AOCM boren@randvax     20 May 86 EASY MARGARITAS
  3 TRUFFLES-MILO  D  AOCM milo@glacier      22 Feb 44 MINDERBINDER
  4 MUSKRAT-1      M  AOCM pjc2@nvuxb.uucp    1 Apr 87 FRED’S MUSKRAT

in /udir/recipes/special/thanksgiving:
  3 CHICKEN-STUFF2 M  AOCM kyrimis@Princeton  8 Sep 87 CHRISTMAS/THANKSGIVING
  4 BUTTERNUT-SQSH C  AOCM russak@pegasus.AT 19 Oct 87 BUTTERNUT SQUASH CASSER
  5 CHESTNUT-STUFF B  AOCM marilyn@merlin.be  8 Nov 87 CHESTNUT STUFFING
  6 PUMPKIN-CAKE-3 D  AOCM mark@megatek.uucp  3 Nov 87 PUMPKIN CHEESECAKE
  7 DARK-ROLLS     B  AOCM marilyn@merlin.be  8 Nov 87 GRAMMY’S DARK ROLLS
  8 CHEESE-SOUP1   SPVAOCM yduJ@edsel.uucp    5 Nov 87 PUMPKIN CHEESE SOUP
```

Figure 7. A partial queue listing

Publication

The weekly publication program is disarmingly simple. It does these things:

- Assembles the ‘boilerplate’ components of the weekly issue.
- Consults a configuration file to discover how many articles to publish, and removes that many from the publication queues to place in the issue.
- Updates the archives to include the contents of this issue.
- Regenerates the various indexes (keyword index, author index, etc.) which are used to speed retrieval from the archives.
- Sends me electronic mail announcing that the issue has gone out, and summarizing what went into it.

Every periodical needs to maintain an archive of its past issues. To make sure that the archive is kept reliably up to date, the filing of articles in the archive is made an

inseparable part of the publication process, and not something that somebody must remember to do by hand every week.

A periodical such as *The USENET Cookbook*, which is devoted to the dissemination of what amounts to a database, must keep not only an archive of all past issues, but also a database of the current set of correct versions of everything that has ever been published. These will differ as soon as a correction has been published: the database must contain the corrected version of an article, while the archive of past issues contains the erroneous version and the correction (or perhaps it contains the bad version followed by the good one, if the correction was a republication).

The past-issue archive is clearly the more general of the two, since the database of published articles can be derived from it, but not vice versa. In the first few months of operation of *The USENET Cookbook*, whenever a correction was needed I published only the correction. This generated a punishing amount of traffic to the back-issue server, as every new subscriber who had begun reading the publication since the original article now wanted, all at the same time, to retrieve the original article so that he could apply the corrections to it. To reduce the amount of traffic to the back-issue server I changed the editorial policy so that whenever a correction was needed, the entire article was republished.

With a policy in place to republish all corrections, the required database can be represented as a directory full of links to the chronological archive. The file named 'african-stew' in the database directory is actually just a link to the most recent archive entry publishing an entry with that name. This scheme creates a slight bookkeeping problem in the event that a database entry needs to be updated (and republished). There is a certain temptation to edit the database entry, then copy it to the publication queue. The problem is that since the database entry is just a link to the archive, then editing the database entry effectively changes history. To prevent this kind of thing, the database and archive are represented with write-protected files, and editorial staff must be trained not to try to override it, lest the archives themselves be modified.

The mechanized proofreader is an *awk* script[6] and simple parser which reads each article to compare it against a simple grammar, and also makes a pass through the article checking for certain kinds of consistency. For example, it checks to make sure that all identifiers are 14 characters or shorter (to accommodate some variants of Unix in which file names are limited to 14 characters). It checks to make sure that all recipe ingredients in the 'list of ingredients' section of the recipe are mentioned in the narrative instructions. The reverse check, that all ingredients mentioned in the narrative are also in the ingredient list, is much more difficult, because the identification of ingredient words in running prose requires a natural-language-understanding program. It checks the computer mail address of the article's author to make sure that it is syntactically correct, and gives a warning message if the address appears to be semantically incorrect. It checks to make sure that all U.S. units of measurement appearing anywhere in the recipe are accompanied by a corresponding metric equivalent. (It does not check that the metric equivalent is correct, because the definition of 'correct' is too complex.) And so forth. The proofreading program checks about 75 different details of the article, and reports error messages on its standard output. The proofreading program is called from an Emacs key binding program, and if that program sees that the proofreader has approved the article, then it (the Emacs program) edits an 'Approved:' header line into the text of the

article and adds a copyright notice to it. Once the 'Approved:' line is in the header, the copy editor is permitted to copy the article to a publish queue.

Subscriber software package

Subscribers who do not have Unix systems receive plain-text copies of the articles, and treat them as ordinary text files. Subscribers who have Unix systems can install a copy of the 'subscriber software package,' which enables them to do various useful processing of the received articles. The subscriber software package is published twice a year, and can be retrieved on demand from the back-issue mail response server. It contains these programs:

- **rckeep**: keep one article. Intended to be called from inside the mail or news reading program. This program extracts the article that the subscriber is currently reading and saves it in a database directory.
- **rckeepnew**: keep all new articles. Intended to be run at intervals or from a batch job. It extracts all articles that have arrived since the last time **rckeepnew** was run, and saves them in a database directory.
- **rctypeset**: typeset one recipe and print it on a laser printer or typesetter.
- **rcbook**: typeset the entire contents of the database directory, adding an index, table of contents, and introduction. Print this typeset document on a laser printer or typesetter.
- **rcnew**: typeset all articles that have been added to the database since the last time **rcnew** was run. Make a complete new copy of the index and table of contents. Various subscribers have contributed additional programs which will probably be integrated into a future release of the subscriber software. One subscriber contributed a program that reads a collection of recipes and makes a shopping list. Another subscriber contributed a program that uses clever index files to do a high-speed search for all recipes using a certain ingredient, to help answer questions like "what shall I do with these lamb chops?"

All of the subscriber programs that do typesetting or printing have the ability to print the articles using either U.S. measures or metric measures. A Canadian subscriber submitted versions of the typesetting program which print both sets of measures, side by side. I have not included these as part of the subscriber software package because the two versions of the recipe are not always identical: sometimes the U.S.-measure version of a recipe will make a different amount of the dish than the metric-measure version does, because of differences in customary packaging. For example, U.S. grocery stores often have 12-ounce packages of ingredients, while grocery stores in Europe would carry 250-gram and 500-gram packages of the same ingredient. The metric version of a recipe is arranged to use 250 grams of the ingredient, with correspondingly smaller amounts of other items, while the U.S. version of the same recipe will call for 12 ounces, which is about 350 grams.

Mail-response server

The mail-response server is a collection of about 12 programs which together manage the back-issue and subscription department of *The USENET Cookbook*. The entire server

runs as low-priority background jobs on my office workstation; it runs at a low priority so that its operation will not interfere with whatever other work I am trying to do. If the load became a burden, it could be run on a server or timesharing machine, of course, but I like to keep an eye on its operation so I run it on my Microvax II.

When an incoming message arrives, it is given a serial number and placed in a work queue. Every few minutes, a daemon job starts, which processes the work queue sequentially. This queuing ensures that no more than one request will be processed at a time, thereby leveling the load somewhat.

The first step in processing a message is to determine its return address. This is a black art and it can never be done perfectly. Once the best guess at the return address has been generated, the body of the message is converted to lower case, and all lines that do not begin with a recognized keyword are discarded. It is then passed to a parser, which decomposes it into a request type and an argument list. There is a separate program to handle each type of request. Requests for information are handled immediately, and the reply is sent back by return mail during the processing of the request. Requests for large amounts of data are queued for later delivery. When a request is queued, the subscriber is sent an acknowledgment that his request has been received, and an estimate of how long it is likely to sit in the queue.

The delivery queues are drained at a regulated rate of 100 000 bytes per hour during the day and 500 000 bytes per hour from midnight to 8am. This prevents the communication lines in and out of our facility from being overwhelmed by surges in the subscriber demand for data. Naturally these numbers are adjustable, and from time to time I tweak them up or down to compensate for varying conditions of our communication networks. There are also safety valves which shut down the entire queue-processing systems in the presence of certain kinds of local network failures, to prevent logjams of various kinds. For example, if the outgoing mailer's queue ever exceeds 100 messages, for any reason, then all server queue processing is suspended.

The server has processed 5484 requests in the last 90 days; the average size of each request was 5 items. This is an processing rate of 300 items per day, which is well beyond the capability of one part-time person to handle manually. The breakdown of request types and disposition is as follows:

53%	Successful request for recipes
20%	Successful request for index
8%	Request for help
7%	Request for a copy of the subscriber software
5%	Recipe request with some items requested unknown
3%	Message contained no comprehensible command
2%	Recipe request with all items requested unknown
1%	Request for the entire contents of the recipe database
1%	Request containing only blank lines

In general the requests with 'all items requested unknown' come from people who have not yet figured out how to use the server and who are experimenting with asking it for nonsense. The requests marked 'message contained no comprehensible command' are almost invariably requests for recipes written in an informal English style rather than in the required server command language; these probably come from subscribers who fail to realize how stupid the server software is. In the last 90 days, 49 requests have arrived asking for every recipe that was in the database at that time.

Obligatory software magnitude statistics

It is obligatory in all writing about software systems to spend a certain amount of time discussing the size and shape of the software itself.

The software for *The USENET Cookbook* is written almost entirely in Unix */bin/sh* shell scripts and in *awk*. The only exception is a program which does date parsing and date arithmetic; this was written in C.

The subscriber software is 9 shell scripts whose total size is 784 lines. They are written in an extremely conservative subset of */bin/sh*, in order that they can run on as many different subscriber computers as possible.

The archive server is 1000 lines in 11 shell scripts with some embedded *awk* code. Most of the complexity of the archive server is in the scheduler, which is 150 lines.

The publishing software is 32 shell scripts and one 200-line C program. The 32 shell scripts total 1300 lines, of which the proofreader is by far the largest (300 lines). The 31 remaining scripts average about 30 lines each.

PROJECT STATUS AND FUTURE DIRECTIONS

This section elaborates the history and status of *The USENET Cookbook*, and reflects on future directions for it and for similar systems.

History

The first issue of *The USENET Cookbook* was published on November 30, 1985. It is now published in the USENET newsgroup named 'alt.gourmand'. The USENET readership measurement figures show that it has about 9000 readers; there are an additional 900 names on the mailing list. These figures probably do not indicate the number of readers, but rather the number of sites at which there is a reader, because the work involved in maintaining the subscriber software is high enough that there is no point in having more than one person do it at a given installation. People seem to share copies of the database files.

As of December 7, 1987 there have been 442 articles published in 99 issues (for various obscure reasons publication was suspended from 25 April 1987 to 25 June 1987). This is an average of 4.5 articles per issue. In the beginning, the limiting factor in the number of articles per issue was the number of articles that were submitted, but in the steady state the limiting factor is the number of articles per week that I can process. Although the editorial work process is set up to permit several people to share the work, I have been doing all of it myself, primarily because I want to keep a complete overview of the entire electronic publication process so that I can spot the need for changes and make them.

The archive directory contains 1.72 megabytes of archives; this number is, of course, steadily increasing with time. Each passing month brings a number of new subscribers, and many of them eventually want to acquire all of the back issues. The load on the mail-response server has not yet gotten out of hand, but unless some technological step is taken in the next year, it will become overloaded.

About 200 different people have submitted articles for publication. Most contributors have sent in only one or two articles, but a few have sent in as many as twenty. I have

rejected about five percent of the submissions as being inappropriate. I should have rejected a higher percentage, because some of the early marginal-quality recipes give unfortunate examples to would-be authors about the kind of article that will be published.

The medium of computer mail makes it much easier for angry writers to harass the editor about their displeasure at his having changed their words. One author submitted a recipe calling for '15.5 ounces' of canned fish; he sent in a barrage of angry complaints about my rounding that to '1 pound' and '500 grams' for publication. Many people insist on cute or affected writing styles. Others insist on long-winded narratives of what they were wearing when they first created the recipe. Surely conventional publishers receive these same kinds of submissions, but conventional authors do not have the technology to bombard the editor with complaints nearly so effectively as these.

Like all editors, I have my favorite authors; whenever a submission arrives from, say, a certain woman at Hewlett-Packard or a certain man at a Berkeley software company, I find myself giving it preferential treatment. Another frequent submitter, an extremely skilled writer as well as a superior cook, delights in sending me wonderful recipes that are written in such a way that I will find it nearly impossible to rewrite them into the 'house style' without utterly destroying her prose; each new article that she submits is more dramatic than the last.

Plagiarism is rampant. Many recipes that are submitted as 'old family treasures' are in fact taken from Betty Crocker or Mrs. Beeton. Perhaps the submitter's grandmother is the one who originally clipped the recipe, but it is almost always possible to find someone's old family recipe in a standard cookbook. Luckily U.S. copyright laws make it impossible to copyright the formula for a recipe; a recipe's author can copyright the words that he uses to describe his recipe, but not the actual formula.

Future directions

The implementation of *The USENET Cookbook* is certainly a success, but it is not by any means the ultimate such system. It is very limited in a number of ways and very primitive in a number of ways. This section has a few ideas for how to expand, improve, and replace it.

The current set of editorial technology, with editing stations and queues, works well in the name space of a single file system. It would be very interesting to see how well it would work if the various editors were at different locations, separated by communication links of various kinds. A worst-case scenario would be to have one editor in the UK and another editor in California, because the electronic interconnectivity between them is sub-optimal.

The current representation scheme, basing published articles on a primitive dialect of *troff*, has worked well, but it is exceedingly limiting. I want to try a more advanced scheme with a richer representation that permits graphics as well as text. In a few years it will be possible to assume that the vast majority of subscribers will have some capability for printing PostScript files, and it will become worthwhile to engineer a publication system somehow based on PostScript imaging rather than *troff*.

The USENET Cookbook is a special-purpose periodical used to build a special-purpose database on the subscriber's computer. It would be very interesting to experiment with a more general-purpose mechanism and a more diverse subject matter.

The subscriber software must be very conservatively written, so that it will run on as

many different computers as possible. In a few years it will be safe to assume that the vast majority of subscribers have access to the X Window System, and it would then be useful to write the subscriber software to be able to take advantage of an interactive window environment.

The biggest long-term problem for electronic publishing ventures such as *The USENET Cookbook* will be accounting and billing. As soon as money starts changing hands and the issue of protection and copyright becomes important, all of the rules will change. Ordinary periodicals can protect themselves because a copy of a weekly magazine is not as valuable as the original. A copy of an electronic magazine is indistinguishable from the original.

CONCLUSION

Online electronic publication of a periodical whose content is not related to computers is indeed possible. Current (1987) technology is strained in supporting such a publication, and subscribers need a relatively high degree of technological sophistication to participate now. There is a tremendous benefit of uniformity of the subscriber's computing environment: based on my experience, very few subscribers will be technologically able to implement their own subscriber software or even modify released software to work for them. Today's state of the art does not offer enough uniformity of graphics or display interface to make practical an electronic periodical carrying more than just text.

One of my worries upon beginning the experiment was that because the readers of *The USENET Cookbook* were not paying for the service, that the experiment would not be particularly relevant to more traditional periodicals. Surprisingly, the answer is that readers *do* pay for the service, but they don't pay money. The world's electronic networks are so full of junk that every reader who participates in them pays a price of his own time. An hour that a reader spends reading something on a computer network is an hour that he could otherwise have spent playing golf or reading *The New Yorker* or reading *Der Spiegel*. USENET currently averages 80 megabytes of traffic per month; *The USENET Cookbook* averages 0.12 megabytes of traffic per month (about 0.1% of the overall network traffic).

REFERENCES

1. John S. Quarterman and Josiah C. Hoskins, 'Notable Computer Networks', *Communications of the ACM*, **29** (10), 932 (1986).
2. Brian K. Reid, 'USENET news.lists', USENET Readership monthly summary report for November 1987. (published monthly by Brian Reid from DEC Western Research)
3. J. F. Ossanna, 'NROFF/TROFF User's Manual', Bell Laboratories: Computing Science Technical Report 54 (1977).
4. David H. Crocker (Editor), 'Standard for the format of ARPA Internet text messages', *ARPAnet RFC 822*, Network Information Center, SRI International, Menlo Park, CA 94022
5. Richard M. Stallman, 'EMACS: the Extensible, Customizable, Self-Documenting Display Editor', in *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, pp. 147–160, (1981).
6. Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988.