Wiggins, Rich. "The University of Minnesota's Internet Gopher System: A Tool for Accessing Network-Based Electronic Information." The Public-Access Computer Systems Review 4, no. 2 (1993): 4-60. To retrieve this file, send the following e-mail messages to LISTSERV@UHUPVM1 or LISTSERV@UHUPVM1.UH.EDU: GET WIGGINS1 PRV4N2 F=MAIL and GET WIGGINS2 PRV4N2 F=MAIL.

1.0 Introduction

Late in 1991, a new creature began burrowing its way around the Internet. This new critter helps people browse many of the resources available on local campus networks or on the worldwide Internet. Called the Internet Gopher, this tool was developed at the University of Minnesota. Pioneering sites began deploying Gopher servers in December 1991. In the short time since, the Gopher system (henceforth called "Gopher") has been deployed at many institutions around the world. A worldwide community of "Gophernauts" has come into being, with various sites contributing ideas, utility tools, bits of code, and schemes for cooperative registry efforts. Gopher now accounts for a substantial fraction of the traffic on the Internet. Gopher servers are delivering text, index searches, still images, audio, public domain software, and more to users all over the world. With Gopher, a user can:

- Find out what movies are playing in Aachen, Germany.
- Learn what earthquakes took place yesterday.
- Read today's student newspaper from a school 2,000 miles away.
- Pick up a quote from Paradise Lost for a term paper.
- Read the city council meeting minutes from Wellington, New Zealand.
- Listen to the final U.S. Presidential Debate of 1992.
- See what Michigan State's campus looks like in spring.
- Read about the Human Genome Project.
- Learn about Federal grants.
- Download public domain software.

The above examples are a tiny sample of the kinds of information Gopher can deliver. An illustration of the value of Gopher comes from a user who works at Michigan State University:

> I wanted to drop a quick line and tell you how much Gopher means to me. I discovered Gopher about two months ago and

> cannot believe how much information is out there. I have found the new Veronica option very helpful as it allows me to build a directory of items that are specific to my interest. This is undoubtedly a great service for anyone who finds it. However, for me it is unbelievable. I am legally blind and I have always said that the most difficult aspect of blindness is the lack of readily available information. Gopher has the ability to change all of that. For the first time, I feel like I can easily and independently access important campus and worldwide information. . . . I use a speech synthesizer and a PC compatible computer to access the Gopher system.

This article describes the Internet Gopher: why it is needed, how it is used, its genesis and early evolution, the underlying protocol (and the new Gopher+ protocol), Gopher's role as a campus-wide information system (CWIS), and its emerging role as an Internet-wide information system. The article also discusses navigational enhancements (e.g., Veronica), organization and quality of service issues, privacy and security concerns, electronic publishing issues, distribution of multimedia information, related client and network information technologies, and Gopher's future.

2.0 The Internet and the Need for Navigation Tools

Today, many computer users at universities, government agencies, and commercial firms are connected in one way or another to the Internet. The Internet is a worldwide network of networks. Campus Ethernets and other local area networks are connected together by a complex web under the aegis of regional networks, national networks, or in some countries, the PTTs (postal/telephone authorities). The constituent networks all use the TCP/IP protocol family; the result is a worldwide network of computers that can communicate with one another. The Internet evolved from the old Defense Advanced Research Projects Agency network (ARPANET). ARPANET sites consisted mainly of Department of Defense research installations and affiliated research institutes and universities. Many thousands of host computers and user workstations now have Internet connectivity. The number of computer users with Internet access is estimated to exceed 10 million.

As a "network of networks," the Internet can be thought of as the aggregation of various campus, corporate, and other networks. Other than following certain rules known as "acceptable use policies," institutions on the Internet are generally autonomous. Therefore, services that are offered on these campus networks are provided because some local service provider believes it's worthwhile to do so. Often, the motivation is to serve local users. For example, a campus library puts its online catalog on the campus network for the sake of local patrons, not primarily for the benefit of Internet users.

As multiple campuses mounted similar services to satisfy the needs of their own communities, Internet users began to find value in using online resources from other institutions. For instance it might be useful to scan the online catalog of a partner interlibrary loan institution. But without a list of

host names (or IP addresses) of the various online catalogs on the Internet, each individual user has to maintain his or her own listing: picture a wall of Post-it notes listing names and addresses next to a user's PC.

Online catalogs aren't the only list of Internet services a user would want to keep up with; all sorts of services are available on the Internet. Examples include:

- List servers and discussion groups.
- USENET News (a sort of distributed discussion facility).
- Anonymous FTP archives, containing public domain software and other offerings.
- Tools and services for finding people.
- Host computers that require assigned passwords.
- Databases that allow logins via Telnet.
- Databases that support the client/server model (the chief example is WAIS).
- Archives of electronic journals and books--the nascent virtual library.

The need for an "Internet address book" applies to all of these kinds of services. Several tools are evolving to help bring order to the Internet. Various Internet navigation tools have different goals and capabilities:

- HYTELNET provides a hypertext database of online catalogs and other systems.
- Archie serves as an index of FTP sites, identifying sites that hold particular files.
- WAIS allows users to search one or more databases using natural language queries; it presents ranked search results.
- World-Wide Web supports networked hypertext.

All of these tools will be discussed later in this paper. For now, the focus is Gopher.

In a nutshell, Gopher offers access to files and interactive systems using a hierarchical menu system. The organization of the menu is defined by the administrator of each server. The resources that each menu item describes, such as ASCII files, Telnet sessions to other computers, and submenus, are called "documents." The scope of a Gopher server could be limited to a campus or to a subject area. Or, a Gopher server could include a catalog of a variety of Internet resources. The following section depicts a "walk-through" of "Gopherspace," showing how Gopher operates along the way.

3.0 Overview of Gopher from the User's Point of View

Gopher serves as a menu system for networked information. The user connects to one of the thousands of Gopher servers now in production around the world and receives an initial (or "root") menu. When you use a tool like Archie, you are asking the server a specific question (e.g., "Tell me what FTP sites can provide a copy of the program named PKZIP"). When you connect to a Gopher server, you are asking it "Give me a menu listing the resources you have to offer." The menu can include submenus. Each Gopher server presents a hierarchy established by the local administrator.

Gopher follows the client/server model. This model divides the labor between the program the user invokes (the "client") and a program running on a host computer, such as a UNIX workstation or a mainframe (the "server").

It's best to run Gopher with client software installed on the user's workstation because it provides a superior user interface and opens up the world of still images, audio files, and other resources. But a user who has not installed a client can still use Gopher: the developers have provided a sort of central client software, and many Gopher sites offer public client services. The public client software is sometimes known as the "curses" client, after the UNIX screen management tool of that name. Users connect to these public client services via Telnet (or, depending on their local network services, via a VT 100 dial-up session).

The following initial tour of Gopherspace will use sample screens from a public client service. This example will connect to the service offered at the home of Gopher, the University of Minnesota. To connect to the public client service at "Gopher Central," one would type the following:

```
telnet consultant.micro.umn.edu
gopher [Type "gopher" in response to login prompt.]
```

The user's Telnet program must be capable of VT 100-style full-screen operations (virtually all are). The client service will respond as shown in Figure 1.

Figure 1. Choosing the Terminal Type

```
TERM = (vt100)  [Press Enter at this prompt.]
Erase is Ctrl-H
Kill is Ctrl-U
Interrupt is Ctrl-C
I think you're on a vt100 terminal
```

At this point, the user should hit the Enter key. Having made this connection, the user would see the root menu of the University of Minnesota's Gopher service (see Figure 2).

Figure 2. The University of Minnesota Gopher's Root Menu

```
            Internet Gopher Information Client v1.12

               Root gopher server: gopher.tc.umn.edu

 -->  1.  Information About Gopher/
      2.  Computer Information/
      3.  Internet file server (ftp) sites/
      4.  Fun & Games/
      5.  Libraries/
      6.  Mailing Lists/
      7.  UofM Campus Information/
      8.  News/
      9.  Other Gopher and Information Servers/
      10. Phone Books/
      11. Search lots of places at the U of M <?>

Press ? for Help, q to Quit, u to go up a menu       Page:1/1
```

As noted above, Gopher presents a hierarchical menu; titles ending in a slash ("/") are submenus (or "subdirectories" or "folders") that list additional choices. For example, if the user presses Enter while the cursor points at the first menu item, a submenu of resources about Gopher will appear. The user can choose among the menu items by using the cursor keys or by typing in the number of the desired menu item. If the menu is longer than will fit on one screen, the "Page: n/m" field in the lower right corner so indicates.

After selecting the "Information About Gopher" menu item, Gopher responds with the menu shown in Figure 3.

Figure 3. Information About Gopher Menu

```
          Internet Gopher Information Client v1.12

                   Information About Gopher

 -->  1.  About Gopher.
      2.  Search Gopher News <?>
      3.  Gopher News Archive/
      4.  comp.infosystems.gopher (USENET newsgroup)/
      5.  Gopher Software Distribution/
      6.  Gopher Protocol Information/
      7.  Frequently Asked Questions about Gopher.
      8.  Gopher+ example server/
      9.  How to get your information into Gopher.
      10. New Stuff in Gopher.
      11. Reporting Problems or Feedback.
      12. big Ann Arbor gopher conference picture.gif <Picture>
      13. gopher93/

Press ? for Help, q to Quit, u to go up a menu    Page:1/1
```

This menu in the University of Minnesota Gopher server is the definitive place to learn about Gopher. Note that menu items for ASCII text files are listed with a period at the end.

Upon selecting the first menu item ("About Gopher"), the file shown in Figure 4 would appear.

Figure 4. About Gopher File

This is the University of Minnesota Computer & Information Services Gopher Consultant service.

gopher n. 1. Any of various short tailed, burrowing mammals of the family Geomyidae, of North America. 2. (Amer. colloq.) Native or inhabitant of Minnesota: the Gopher State. 3. (Amer. colloq.) One who runs errands, does odd-jobs, fetches or delivers documents for office staff. 4. (computer tech.) Software following a simple protocol for tunneling through a TCP/IP internet.

If you have questions or comments, you can get in contact with the Gopher development team by sending e-mail to:

gopher@boombox.micro.umn.edu

If you are interested in news about new gopher servers and software you can subscribe to the gopher-news mailing list by sending e-mail to:

gopher-news-request@boombox.micro.umn.edu

There is also a USENET news discussion group called

comp.infosystems.gopher

where Internet Gopher is discussed.

If you want to get the most recent releases of the gopher software, you can get these via anonymous ftp from boombox.micro.umn.edu in the /pub/gopher directory.

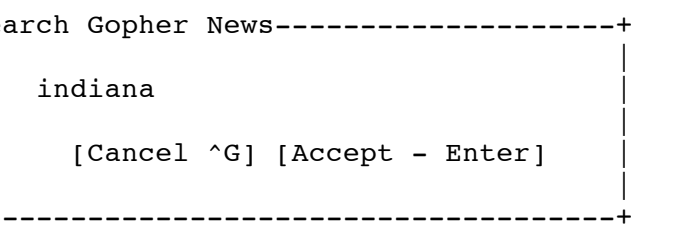
Press <RETURN> to continue, <m> to mail:

In the above example, the curses client leaves the text of the selected text file on the screen until the user types Enter. After that, the "Information About Gopher" menu will be redisplayed. Because Gopher is organized hierarchically, you need a way to move back a level in the directory tree. With the public "curses" client the user simply types "u" for "up." No matter how deep in the hierarchy the user travels, the client is able to back up, one menu at a time, until the root menu is redisplayed.

The "Information About Gopher" menu above included a menu

item entitled "2. Search Gopher News <?>"; this menu item offers an index search, as denoted by the question mark. Gopher servers generally implement indexing using the public domain WAIS (Wide Area Information Servers) search engine. (A common exception: servers implemented on NeXT workstations often exploit the Digital Librarian delivered with NeXTStep.) Upon selecting this menu item, the user is prompted to enter a keyword (see Figure 5).

Figure 5. Search Gopher News



Note that most Gopher clients will highlight the sought keyword within the text of the displayed document. This makes it easy to find the occurrences of the keyword in context.

Searching is not currently as flexible as one would like. In particular, the standard release with the WAIS engine does not provide for Boolean or proximity searches. In November 1992, Don Gilbert of Indiana University announced modifications to the WAIS indexing engine normally used with Gopher servers. His enhancements include Boolean, partial word search, and literal phrase searching. His biology-oriented Gopher (located at ftp.bio.indiana.edu, port 70) allows testing of these search features. Examples of the kinds of searches possible include:

- Boolean: red and green not blue. Result: just those records with both the words "red" and "green," excluding all records with the word "blue."

- Partial words: hum*. Result: all records with "hum" (e.g., "hummingbird," "human," and "humbug").

- Literal phrases: red rooster-39. Result: only those records with the full string "red rooster-39" will be retrieved.

The scope of a Gopher index is determined by the administrator. The administrator can choose to index one file, all files in a subdirectory, or all files in a directory and its subdirectories. Often a large file is broken up into a series of small files so that it can be loaded into Gopher. This will allow the user to selectively retrieve sections of interest. Usually, the wording of the Gopher menu item makes it clear what the scope of a given index is. It's the administrator's job to make sure this is the case.

You can best learn more about Gopher by browsing through various servers: connect to "Gopher Central" (gopher.tc.umn.edu,

port 70) and follow the list of Gophers. Alternatively, you might use the global title index at Michigan State's central Gopher to look up these Gopher servers (or any others of interest) by keyword. (The index of Gopher server names is on gopher.msu.edu, port 70; look under the "More About Gopher/Other Gopher Servers" menu item.) Also, you may want to try Veronica (discussed below) as a way to locate specific Gopher documents.

4.0 Gopher Clients

Recall that the above examples came from using the public client. The best access to Gopher documents requires use of Gopher client software running on the user's workstation. Gopher clients have been implemented on a variety of platforms. The University of Minnesota keeps an archive of commonly used clients, developed there or elsewhere, on its anonymous FTP service, which is located on boombox.micro.umn.edu. Common clients include:

- PC Gopher III--the University of Minnesota's PC client, which was written using Borland's TurboVision. It provides a quasi-graphical interface complete with mouse support. PC Gopher is a relatively large program. Since memory use on networked PCs is tight, the program's size has been problematic.

- UGOPHER--a PC client from the University of Texas Medical School at Houston. It is a port of the UNIX curses client. It provides a very simple interface, but it demands little memory. It supports special data types such as TN3270 and still-image files.

- Novell LWP client--a PC client from the University of Michigan Medical School. This client works with Novell's LAN Work Place for DOS. It supports images and audio as well as TN3270. It sports a friendly graphical interface with more options than the standard client. As of this writing, it does not support a mouse.

- Gopher Book--a Windows client from the Clearinghouse for Networked Information Discovery and Retrieval. Gopher Book runs under Microsoft Windows and implements a book-like view of Gopherspace. The Gopher community has long wanted a good Windows client; this could be it. (Users may want to FTP to sunsite.unc.edu and look under pub/micro/pc-stuff/ms-windows/winsock for Gopher Book and related tools, such as the Winsock DLL.)

- TurboGopher for the Macintosh--a Macintosh client from the University of Minnesota. Various Mac Gophers have been implemented, at the University of Utah, Brown University, and at other sites. TurboGopher appears to be highly functional, robust, and efficient, and it is on its way to superseding the other Mac offerings.

- Curses client--a generic client for UNIX workstations. It is also used at many Gopher sites to provide Telnet access to the Gopher world for users who haven't yet obtained client software for their workstations. Users

installing the curses client on their UNIX workstations must build the client from source code on the target machine (as is commonly true with UNIX software offerings). A version of this client can also be compiled for use under the DEC VMS operating system.

+ Page 15 +

- NeXT Gopher Client--a NeXT client from the University of Minnesota and the University of St. Thomas. This client makes good use of the NeXT's large windows, and it leaves the last two or more menus on the screen, providing useful contextual information. Recent modifications have been implemented by Jim Lebay of Michigan State (icon support, bug fixes, better handling of windows, and support for image files) and David Lacey of the University of Iowa (support for the MIME protocol).

- Xgopher--a client from Allan Tuchman at the University of Illinois that runs under X-based UNIX systems. (The X release must be later than X11 Release 3.) Xgopher supports multiple active items and an easy-to-use graphical user interface. It is highly configurable. A C compiler is needed to build the client for the target user's machine.

- Rice University CMS Client--a client that allows users of VM/CMS mainframes to connect to Gopher servers. The host must have outbound VT 100 Telnet to function well when connecting to Unix-based services. (The IBM 3270 terminal protocol does not lend itself well to outbound connections to byte-oriented hosts; check with local support personnel to determine if such access is available at your site.)

- MVS Gopher Client--a client from Draper Laboratory for TSO users on MVS mainframes. This client can provide 3270 terminals with access to Gopher services.

An experimental OS/2 client has been announced by David Singer of IBM Almaden (look for os2gopher.zip on boombox.micro.umn.edu).

Gopher clients vary widely in their appearance and features. The same documents may be displayed to users in quite different form, depending on which client is used. Ultimately, the information is the same, even though the display format varies. (Marshall McLuhan would have a field day.) Some clients allow the user to print an entire document. With the PC client, the document goes to a locally attached printer; however, the NeXT client assumes its printer is a PostScript device, which might be connected over the local Ethernet. Some clients let the user save a document to the local workstation's disk; others allow the user to send a document via e-mail to the destination of choice.

+ Page 16 +

Gopher clients need a way to display files on the user's screen. This can be done by code within the client program itself. Alternatively, the client may launch an external tool, often called a "browser" or a "pager." For instance, UGOPHER has a relatively limited built-in browser. The user can install a

superior file display tool, such as the popular shareware tool LIST from Vernon D. Buerg, and tell UGOPHER to use it instead.

Similarly, clients may launch separate programs to open Telnet sessions, do CSO searches, play audio files, and so forth. The user must install all the needed external tools and configure the client to use them. Installation instructions are provided with every client.

A useful feature in many clients is the "bookmark." Upon finding an item of particular interest, the user sets a bookmark. The client software stores the bookmark on the client workstation's disk. In a later session, the user can call up the list of bookmarks and immediately jump to items of particular interest without having to navigate the menus. Because resources of interest could be buried deep within Gopher servers, the bookmark option lets the user build a customized view of Gopherspace. (Some users have suggested the need for hierarchical bookmarks; the current bookmarks provide a simple flat list.)

Most clients also have the ability to save documents themselves on the user workstation. Some information providers have found that Gopher makes a very efficient mechanism for delivering files to users. At least one software archive encourages its users to obtain software via Gopher instead of anonymous FTP. (The Stanford University Macintosh archive, Sumex, suggests users choose Gopher over FTP or other methods.)

Gopher was originally written with the assumption that the user would be resident on the Internet, with a permanently assigned IP address, and have client software on his or her workstation. This model does not always provide dial-up users (users dialing in over a phone line using asynchronous ASCII terminal emulators) with the same level of service as on-campus users receive. Many institutions are experimenting with schemes to allow dial-up users to appear to be on the Ethernet, using protocols such as SLIP (Serial Line IP) and PPP (Point-to-Point Protocol). Under SLIP or PPP, the user can run a standard Gopher client, which works as if it were on a local TCP/IP network. (Of course, data is delivered more slowly; even today's high-speed modems don't match local area network speeds.) Because these dynamic IP assignment schemes are relatively new and they are not widely used, traditional dial-up users must usually rely on the public curses client.

In general, users who run Gopher clients benefit from a superior environment and have access to more data types. However, recent versions of the public curses client make it possible for users to download files for viewing outside of their terminal session. To download a file, one strikes a "D" (that's a capital "D"; use lower "d" at your peril, for that says you want to delete a bookmark) with the cursor on the document title. When this option is invoked, the public client service asks the user which protocol to use (i.e., raw text, Kermit, XMODEM, YMODEM, or ZMODEM). Assuming the user's terminal emulator can handle one of these protocols, this allows the dial-up user to obtain a copy of a file, such as a still-image file, that otherwise could not be displayed using the curses service.

5.0 Obtaining a Gopher Client Via FTP

A common way to obtain a client is by use of anonymous FTP. The

University of Minnesota's software archive resides on boombox.micro.umn.edu. Other sites also maintain their own archives, which may include local fixes or enhancements. In particular, sites may configure clients to point to local Gopher servers by default, or they may make changes to allow the clients to work properly with the local network environment. New Gopher users should consult with local computer support staff to determine where best to obtain a client.

The following is an example of obtaining the University of Minnesota's client for MS-DOS (see Figure 6). It assumes that the user has a copy of PKZIP, a popular file compression shareware tool.

Figure 6. Example FTP Session to Download a Gopher Client

```
C:\GOPHER: ftp boombox.micro.umn.edu
Connected to boombox.micro.umn.edu.
Connected to boombox.micro.umn.edu.
220 boombox FTP server (Version 4.1 Tue Apr 10 05:15:32 PDT 1990)
ready.
Name (boombox.micro.umn.edu:rww): ftp
331 Guest login ok, send ident as password.
Password:wiggins.msu.edu
230 Guest login ok, access restrictions apply.
ftp> cd pub/gopher/PC_client
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls (0 bytes).
total 1352
-rw-r--r--   1 bin          385 Apr  1 15:43 00readme
-rw-r--r--   1 bin            0 Mar 22 17:29
FTP_THESE_FILES_IN_BINARY_MODE
-rw-r--r--   1 bin        75376 Apr  1 15:43 bmkcvt.exe
-rw-r--r--   1 bin         2151 Apr  1 15:43 bmkcvt.txt
-rw-r--r--   1 bin          370 Apr  1 15:43 gopher.bmk
-rw-r--r--   1 bin       182910 Apr  1 15:43 gopher.exe
-rw-r--r--   1 bin        75711 Apr  1 15:43 gopher.ovr
drwxr-xr-x   2 bin          512 Mar 22 17:29 icky_old_client
-rw-r--r--   1 bin          643 Apr  1 15:43 manifest.101
drwxr-xr-x   2 bin          512 Mar 22 17:29 packet_drivers
-rw-r--r--   1 bin        41929 Apr  1 15:43 pcg3.txt
-rw-r--r--   1 bin        62341 Apr  1 15:43 pcg3.worddoc
-rw-r--r--   1 bin       211699 Apr  1 15:43 pcg3.zip
-rw-r--r--   1 bin         2999 Apr  1 15:43 release.101
226 Transfer complete.
838 bytes received in 0.31 seconds (2.66 Kbytes/s)
ftp> bin
200 Type set to I.
ftp> get pcg3.zip
200 PORT command successful.
150 Opening BINARY mode data connection for pcg3.zip (211699
bytes).
226 Transfer complete.
local: pcg3.zip remote: pcg3.zip
211699 bytes received in 6 seconds (34.47 Kbytes/s)
ftp> quit
```

221 Goodbye.
C:\gopher: pkunzip pcg3

+ Page 19 +

At this point, the client software is on your PC's disk. Before you can use it, you must configure it. This includes specifying the IP address of your workstation as well as your local "netmask." If the correct answers for these values are not obvious to you, contact local computer support for assistance. Once you have configured the client, you can invoke it by typing "gopher." Future sessions do not require configuration (unless you want to change a setting, for instance to choose a different Gopher server).

Unfortunately, use of campus or corporate computer networks is not as simple as plugging a cable into an outlet. There are many local variations. Some sites use public domain TCP/IP packages; some have site licenses for commercial products. Some departments have local area networks that are gatewayed into the larger Internet environment. Technical support varies based on computing platform. New Gopher users should consult local network support specialists for assistance.

6.0 How the Gopher Protocol Works

The Gopher protocol rests on the metaphor of a file system. As we've seen, a Gopher server delivers a menu in the form of a list of menu items. When a Gopher client connects to a server, it opens a TCP connection to a specified "well-known port"--typically port 70. Once the connection is established, the client then transmits a "selector string" that specifies what it wants to see. If it is the initial connection to a Gopher, the client sends a null selector string. This tells the server to deliver the highest level, or root, menu to the client. Once the server has finished sending the list of items, the connection is closed. The client then displays the document titles on the user's console, and the user is free to click on items of choice.

The information sent by the server includes the following fields:

> <type> <Document Title> <selector string> <server domain name> <server port number>.

+ Page 20 +

One line of this form is delivered for each document. The initial field is a one-byte document type identifier. The type field is concatenated with the beginning of the title field; other fields are separated by the ASCII tab character. The Document Title field is the descriptive text that the client should display for each item. The "selector string" is a string of characters, usually derived from the location of the document in the server's file system, that can be used to uniquely identify the document for retrieval. The server domain name is simply the domain address of the server. The port number is a concept common in TCP/IP server nomenclature; the Gopher server "listens" on a specified port for transactions. (The Internet Assigned Numbers Authority, or IANA, which concerns itself with such issues, has assigned Port 70 as the standard Gopher port, though a given server may use another port--or ports, if a single

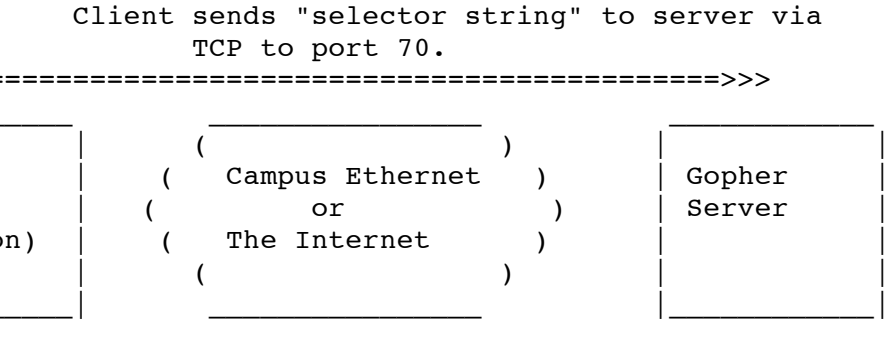
machine runs multiple Gopher services.)

For each document descriptor delivered by the server, the client inspects the one-byte type designation. If a document is of a type the client can't handle, the client simply omits that document from its list of titles. For instance, audio files are not currently supported via PC Gopher III. If a user points PC Gopher III at a directory that contains such files, those titles will not be shown to the user. Since the user can't select it, there's no frustration with impossible requests. (Although the protocol specification calls for this behavior, some clients, such as the public curses client, do not in fact omit such items. This may be useful in some cases. For instance, the user may want to download an item via the curses client for later use outside the Gopher session.)

The process used by Gopher clients and servers can be envisioned in Figure 7.

+ Page 21 +

Figure 7. Gopher Client/Server Interaction



The selector string tells the server what the user wants to see. The delivered document selectors, in turn, are the unique identifiers needed to cause the server to deliver each upon request. Once the server has delivered a document (whether it be folder, plain text, or otherwise), it has done its job for this transaction, so it disconnects. The Gopher server does not retain any information about the client across transactions--it is said to be "stateless." This aspect of the Gopher design is the key to Gopher's efficiency--the server is only connected to the user long enough to serve a particular request, and it does not pay the high overhead cost of having hundreds or thousands of users "logged in" at once. This highly efficient model allows relatively small workstations to function as Gopher servers, handling millions of requests per week from thousands of users across the Internet.

+ Page 22 +

7.0 Gopher Document Types

The Gopher document types that have been defined so far are shown

in Table 1.

Table 1. Gopher Document Types

0	File
1	Directory
2	CSO phone-book server
3	Error
4	BinHexed Macintosh file
5	DOS binary archive
6	Item is a UNIX unencoded file
7	Index-Search server
8	Text-based Telnet session
9	Binary file
+	Redundant server
T	Text-based TN3270 session
g	GIF format graphics file
I	Image file

Source: Internet Request For Comments document, RFC 1436.

In practice, other document types have also been adopted (e.g., "M" has been used for MIME mail documents).

As Table 1 illustrates, the term "document" is used broadly to include any type of resource that can be accessed by the Gopher. Here is a more detailed explanation of some of the important document types.

- File. This is a simple ASCII text file, which is displayed on the user's workstation using some sort of file browser.

- Directory. This is a list of documents that is used to construct a Gopher menu. When a directory item is selected, the server sends the client the list of items in that directory. Included with each item is the information that the client will need in order to fetch the document when the user requests it.

- CSO phone book server. Named after the Computing Services Organization at the University of Illinois, CSO provides a client/server protocol for searching a phone database. Gopher recognizes this protocol and lets the user interact with a CSO server in order to look up information.

- Text-based Telnet session. This document type allows a Gopher to present a list of host services that accept Telnet as a remote access protocol. For instance, a list of Internet-accessible online catalogs.

- Text-based TN3270 session. A variant of Telnet, TN3270, is required to connect to IBM mainframe hosts. Support for this form of connection was incomplete early in the life of Gopher but has become pervasive in the last several months. (TN3270 support is important

because many online catalogs and other database services reside on IBM mainframes. These days, when traditional mainframe-based services are looked upon with derision by some in the networked information community, mainframes are sometimes referred to as "legacy systems." But a lot of valuable information is still stored on those legacy systems, so connectivity to them is necessary.)

There are also document types for PC (DOS binary archive), Macintosh (BinHex file), and UNIX (unencoded file) files; graphic files (GIF); searchable databases (index-search server); and backup Gopher servers (redundant server).

Note that the case of the document type identifier is significant. Because each document descriptor line contains the name of the server where that document is located, it is easy for a Gopher server to point to documents stored far and wide. For instance, a Gopher server at Mythical State University might set up the document types for a root menu as shown in Table 2.

Table 2. Example Document Types

```
Type: 0
Document Title: About this Gopher
Selector: 0/about_MSU
Server: gopher.mythical.edu
Port: 70

Type: 1
Document Title: Fun & Games
Selector: 1/fun
Server: gopher.tc.umn.edu
Port: 70

Type: 1
Document Title: MSU Campus Events
Selector: events
Server: events.mythical.edu
Port: 70

Type: 2
Document Title: MSU Telephone Directory
Selector: [blank]
Server: cso.mythical.edu
Port: 105

Type: 7
Document Title: Search MSU Gopher
Selector: titles 7/ts
Server: gopher.mythical.edu
Port: 70
```

Note that the Document Title is the only field that clients display by default. The combination of Selector, Server, and Port describes the document uniquely. Note also that the

selector string consists of whatever text the server wants to receive in order to deliver a document. For Unix-based servers, this string is prefixed by the type of the document and a slash. Finally, note the variety of servers shown in this example. Often a Gopher administrator will store items peculiar to his or her domain on the same server machine as the Gopher server, but this is not essential. It is common for documents of local interest to reside on several servers, as shown above. Theoretically, a server could offer only documents that reside on other Gopher servers.

+ Page 25 +

Because the Gopher design calls for a simple protocol built on TCP/IP, it is possible to test the behavior of servers without even using a client. The reader might want to try connecting to a Gopher server "manually" to see how this works. For instance, if you were to open a Telnet session to "gopher.micro.umn.edu 70" and then press the return key, you would see the initial menu for the main University of Minnesota server displayed in raw form.

Most Gopher clients provide an "Item Info" option that displays the selector string that pulls up the current document. This can be useful when you want to know where a given document originates. It also makes it easy for a Gopher administrator to add a local pointer to a newly discovered, useful item.

8.0 Setting Up a Gopher Service

It is relatively easy for a developer to implement Gopher client or server software. The protocol is also very efficient in its demands on the server and the network. This simplicity extends to establishing a new Gopher service: it is also easy for an information provider to set up a Gopher server. Some Gopher administrators report being able to bring a server online within an hour or two of downloading the server installation kit. Typical Gopher servers are UNIX workstations of one sort or another. The University of Minnesota relies upon Macintoshes running A/UX and NeXT workstations for the most part. Other popular server platforms include Sun workstations and IBM RS/6000s. The Gopher server code has also been implemented on the PC, but this platform is relatively uncommon as a server. Servers have even been implemented on IBM mainframes, both under the VM/CMS and MVS operating systems.

+ Page 26 +

The standard Unix-based server code is written in C. It can be found at the same repository as the client code (boombox.micro.umn.edu), and it can be installed on a variety of platforms with little modification. The Frequently Asked Questions (FAQ) file often includes tidbits on peculiarities of installation on particular platforms. The news group (comp.infosystems.gopher) also contains discussions of installation issues. New server administrators will want to consult the news group archive (see Figure 3).

Numerous tools have been created that assist in managing Gopher servers. For example, tools that analyze logs, improve indexes, and extend support delivery of calendar-based information are available. Some of these tools can be retrieved from the University of Minnesota's FTP server. Others, particularly short Perl scripts, are posted by the authors on

comp.infosystems.gopher. Thus, the discussion group is not only a source of accumulated wisdom. It also is a repository of helpful tools.

Gopher administrators announce new servers on the following lists:

gopher@ebone.net (European servers)
gopher@boombox.micro.umn.edu (All other servers)

The announcement should include the name of the Gopher server, its Internet address, and its port number. The name can include labels such as "(experimental)" or "(Under Construction)" as appropriate. New administrators may want to review Gopher server names listed in "All the Gopher servers in the world" to see what sort of names others have chosen before picking their own.

Gopher administrators face many challenges. One of these is how to effectively organize the Gopher menu hierarchy. Another challenge is how to convert documents from whatever format the owner of the information prefers to flat ASCII, which is currently the least common denominator document format--the format that all computing platforms can cope with. Prospects for delivering documents in PostScript are discussed below; in the absence of that sort of advance, perhaps the best advice for the Gopher administrator is to insist that document owners do the translation to flat ASCII, rather than taking on that chore as a part of running a Gopher service.

+ Page 27 +

9.0 Gopher's Origins

Computer Center staff at the University of Minnesota began discussing the need for something like Gopher in early 1991. They wanted an online mechanism for "publishing" hints on computing services at the university. Mark McCahill, project leader for the group that developed Gopher, says the goal was to find a scheme that was more efficient than one-to-one consulting or conventional handouts and short courses. At the time, another group at the university was proposing a mainframe-based solution for campus document delivery. The group designing Gopher wanted a simple, network-based scheme that supported browsing of available documents. They also wanted to build a service that was fun to use, so that a critical mass of users would develop. The original Gopher team included Farhad Anklesaria, Paul Lindner, Daniel Torrey, Bob Alberti, and Mark McCahill.

The developers' earliest discussions produced a goal of a simple protocol--easy to understand, describe, and implement. They decided upon a client/server model: a client would open a Telnet connection to the server, request specific information, receive and display the information, and disconnect. The initial model called for only two document types: text files and folders (subdirectories). Thus, from the beginning, the developers envisioned a mechanism that would deliver lists of files and directories upon request. The client/server model would provide for sessions lasting only long enough to deliver that list to the user's client software. These aspects of the Gopher model have endured.

The University of Minnesota design team held their first serious meetings during the first week of April 1991. The Gopher team proceeded to work on implementing the first servers and clients. The goal at this point was to build a working prototype

that could readily be discarded; the team assumed that whatever they produced might be replaced by something better within a year. The team spent quite a few 16-hour days on the project. By the last week of that month, they had prototype Gopher clients and servers working. Initially, there was a Macintosh client and server, a UNIX client and server, and a PC client.

By late April, the Gopher developers decided that some sort of search mechanism was also needed. NeXT workstations were seen as an appropriate choice for servers, because the built-in Digital Librarian offered a highly functional search engine. The developers also decided Gopher should support telephone directory searches, and they settled upon the CSO telephone directory search protocol, already in use at numerous universities, as the best way to implement online phone books.

These different document types--text, folders, and CSO searches--would be sufficient to define a Gopher that could deliver documents stored on a single server. But, from the start, the protocol allowed a server to point to another server as the physical home of a given document. This allows a Gopher to include services that may be scattered across the Internet all in one list. It also makes possible a directory of other Gophers, any of which is a keystroke or a mouse click away.

But the developers thought about providing ways to connect to existing database servers on campus, such as a university's online catalog. They decided to include a document type that was itself a Telnet session to another host computer. To round out the collection, a document type for transferring Macintosh or PC binary files was included. The original protocol was designed to be extensible; the one-byte data type field could potentially support 255 different data types. A draft Gopher protocol specification was released in spring 1991.

The original Gopher team divided their development efforts. Torrey implemented the client for MS-DOS, and Bob Alberti wrote the first UNIX curses client. Anklesaria wrote the initial server and client for the Macintosh. Lindner developed the initial UNIX server code. Besides serving as project leader, McCahill set up the first index server using the Digital Librarian tool on the NeXT and assisted with early development of the Macintosh client. Despite this division of labor, the group worked as a team on problem solving and common issues.

Delivery of sound via Gopher was born during the early development phase. One weekend the team member doing most of the server code, Paul Lindner, wanted to hear some music in his office, which is several rooms away from the communal CD player. His NeXT workstation was capable of playing sounds, and there was a CD player available on a workstation in another room. A few hours later he had implemented the sound data type, and Gopher was capable of playing music across a real-time Internet link.

The first Gopher production services were in place at the University of Minnesota by late summer of 1991. Gopher was announced to the world via the campus-wide information systems mailing list (CWIS-L@WUVMD) in July 1991. A USENET news group, alt.gopher, was started. Computer system administrators from around the Internet began to learn about Gopher. The code was made available for others to try, and Gopher servers began popping up in various places. By early 1992, Gopher was no longer a prototype but was becoming a tool of choice as universities sought ways to implement campus-wide information

systems. Steve Worona of Cornell Information Technologies says that Cornell was about to design a protocol that would allow them to move their pioneering CUINFO mainframe campus information service to a network/workstation environment, "but then we found out about Gopher, and it was exactly what we were looking for." [1]

+ Page 29 +

The new TurboGopher client for the Macintosh provided a learning experience for the developers, because much of the testing took place over 2400 bps dial-up SLIP connections. This slow link exposed the ways in which the client blocked efficient transfer of data. The client was modified, and it now offers very impressive performance on high-speed networks. The University of Minnesota team believes this to be the fastest client now in service. Recent work on TurboGopher has been done by University of Minnesota staffer Dave Johnson; additions include foreign language support.

10.0 Gopher+

In August 1992, the regional computer network CICNet sponsored a Gopher Workshop in Ann Arbor, Michigan. Invited attendees included Gopher implementers at the various CICNet institutions as well as Gophernauts from Brown, Yale, Cornell, Princeton, Rice, the University of Washington, the University of North Texas, and other institutions. Mark McCahill and Farhad Anklesaria attended from the University of Minnesota. At that conference, the University of Minnesota developers presented Gopher+, their vision for how to extend Gopher beyond its original design.

The original Gopher design, with its one-byte document type, was adequate to meet many of the needs of the community. But there were many demands for additions to the protocol, to handle everything from PostScript files and various image files (e.g., GIF and JPEG) to global document attributes, such as author name and document expiration date. Rather than embed each and every requested extension in the protocol, the University of Minnesota team devised a mechanism to support general ways to add them to the protocol. McCahill says that the team had been working on Gopher+ throughout 1992, but the advent of the workshop caused their ideas to gel.

Gopher+ adds new, named fields to the simple one-byte item descriptor in basic Gopher. If a client appends an exclamation mark to a selector string, the Gopher server is expected to deliver an Attribute Information block. The first named field, +INFO, is required; it resembles the descriptive line sent by the old protocol. Other named fields that have been suggested for Gopher+ include +ABSTRACT, which would be a textual abstract describing the document; +ADMIN, which would be the name of the owner of the document; and +DATE, which would be the date the document was last modified. These global document attributes would help administrators maintain and describe documents, and, after appropriate client enhancements, would help users select documents of interest.

+ Page 30 +

The University of Minnesota announced it would act as a central registry of Gopher+ data types; anyone proposing to

implement a new named attribute field will submit it for registration to the Gopher team. This model allows cooperative uses of new fields as agreed to by the Gopher community. It also potentially allows a given site to implement a particular field in its clients and servers that would be of no interest to anyone else. As always, a client would only handle the information it knows how to deal with. If someone adds a +HAIRCOLOR attribute, a Gopher+ client would be free to ignore it. Note that a Gopher+ field could be a simple textual value, or, like a document under the basic protocol, it could point to another Gopher+ server.

Besides providing a mechanism for supporting global document attributes, Gopher+ is intended to support alternate views to a document. For instance, a special named field called +VIEWS will support versions of a document in different languages. With +VIEWS a document could be offered in a variety of formats, from flat ASCII to PostScript.

Beyond the global attributes and alternate views functions, Gopher+ also provides a mechanism for interactive queries. A Gopher+ server can interrogate a user for specific information such as a password, or it could even serve as an interface between a user and an interactive process on another host. This +ASK support includes options for prompting for file names or for the user to make a choice among a range of options.

In February 1993, the University of Minnesota Gopher team announced the first clients implementing the Gopher+ protocol--a version of TurboGopher (for the Mac) and a version of the UNIX client. Server code is also available, and the production version of the University of Minnesota's Gopher points to a demonstration Gopher+ server. Users with non-Gopher+ clients can safely point to Gopher+ servers, but they will not be able to view the Gopher+ documents.

There has been some criticism of the Gopher+ protocol extensions. In particular, some have argued that instead of defining the +VIEWS scheme under Gopher+, the Gopher community should rely upon MIME (Multipurpose Internet Mail Extensions). Originally proposed as a way to allow arbitrary 8-bit files to survive transit over Internet (SMTP) mail, MIME is being deployed widely and may serve as a standard way to handle multimedia files on a variety of platforms, whether the files are delivered via e-mail or otherwise. Others question whether there really needs to be a Gopher+ registry. If it does need to exist, some argue it should be at the official Internet registry, the IANA.

In April 1993, the University of Minnesota sponsored a second Gopher Workshop and Conference. Participants at the Workshop urged the developers to consider adoption of the MIME content types and registry scheme instead of "rolling their own" types and registry. The Gopher team agreed to "strongly consider" use of MIME content type attributes.

In the short term, the University of Minnesota intends to act as the registry for other global document attributes (such as the owner of a document), but they have stated their willingness to use IANA eventually.

Whether Gopher+ is deployed in its current form or with changes to the syntax and registration process, it is clear that it will allow Gopher to be extended to handle document types that have not yet been envisioned. It will improve the ability of Gopher administrators to organize documents. It will also allow basic improvements in the technology, providing a framework for

improved navigation and interfacing to interactive services. An RFC describing the base Gopher protocol has been issued (RFC 1436). Mark McCahill says that, after minor corrections are made to that document, the base protocol will be frozen. Gopher+ is considerably more fluid. It will be interesting to watch its evolution.

Along with Gopher+, the University of Minnesota team has also developed a mechanism for allowing "lightweight" authentication of users for access to protected documents. For instance, some vendors may insist that their documents can only be read when users have typed in a unique password assigned to them. The AdmitOne Authentication scheme lets a user obtain a "ticket" that allows access to restricted documents. The AdmitOne scheme uses encryption to avoid sending passwords over the network. It also provides a way that a client can reuse a ticket for subsequent transactions. Critics of AdmitOne have pointed out schemes for defeating the security of AdmitOne, some rather elaborate. One claim is that security cannot be provided by a client and a server alone--a trusted third party is required.

11.0 Organizational Issues

The greatest strength of Gopher--its ability to easily present a single menu whose constituent documents reside far and wide on the Internet--also presents some interesting challenges. Gopher is being used at many universities to implement campus-wide information systems (usually referred to by the acronym CWIS, pronounced "kwis"). Each site setting up a CWIS under Gopher faces the challenge of devising an organizational scheme that embraces all the local documents and host services of interest as well as the many documents and services available over the Internet.

Some sites have adopted a simple model for the root menu. This yields a short and simple initial menu, such as:

About Gopher
The Campus
The Community
The World

This elegant approach is appealing in its simplicity. Of course, to some extent, the scheme simply moves the problem of where best to put various documents downstream. Even with these simple categories, some items can be hard to place. Where do you put a gateway to your student e-mail service? What if you have an events calendar that integrates campus and community happenings? Where does the local weather go?

At the other end of the spectrum, a site might choose to have a broad set of offerings on the root menu, making more documents accessible with fewer mouse clicks. Such a menu structure might look like this:

1. Gopher at Michigan State University.
2. More About Gopher (Documents & Navigation Tools)/
3. Keyword Search of Titles in MSU's Gopher <?>
4. About Michigan State University/
5. MSU Campus Events/

6. News & Weather/
7. Phone Books & Other Directories/
8. Information for the MSU Community/
9. Computing & Technology Services/
10. Libraries/
11. MSU Services & Facilities/
12. Outreach / Extension / Community Affairs/
13. Network & Database Resources/

+ Page 33 +

This sort of scheme tries to bring commonly accessed documents to the front. The user need not search through multiple menus to find today's weather or the campus phone book. The very first item is a text file with introductory material, so the new user doesn't face a menu full of folders. A keyword title search allows users to find documents no matter where they are stored in the hierarchy. On the other hand, the user confronts a much busier initial screen, which may be daunting in its length and complexity. Depending on the client's default window size, the initial screen may not even fit on the menu window, forcing the user to scroll to see all choices.

Users may find the bookmark facility offered by most clients to be a useful way to customize their view of a Gopher. Gopher administrators can also make liberal use of indexes--both document title indexes and full-text indexes--in order to make it easy for users to quickly locate data without having to hunt through the hierarchy. These tools can help, but a balanced hierarchy that reflects local needs is a worthwhile goal, even though there is no universal agreement on basic issues such as whether a depth-first or breadth-first organization is best. In any case, the process of building a CWIS under Gopher is much more likely to succeed if a campus-wide committee helps design the structure. For instance, a site might want to include staff from the central computing organization, the library, university relations, departments that run their own Gophers, and so forth. A committee can help ensure needs of various campus constituencies are met.

While an organizational scheme for a CWIS must address the peculiar needs of each campus, most Gophers also allocate part of their directory tree to "Internet Resources" (or some similar name to serve the same purpose). Gopher administrators are beginning to realize that it does not make sense for each of them to come up with a local scheme for organizing all the online resources of the Internet. First, this is not practical; second, if a common organizational scheme can be devised for all of Gopherspace, then users will not confront wildly differing Internet directory structures as they move from campus to campus.

Note that not all Gophers aspire to the scope of a campus-wide system. In some cases, the documents on a Gopher are mostly related to the special purpose of the sponsoring institution (e.g., the Electronic Frontier Foundation, the Well, and the National Institutes of Health). Such Gophers are inherently more narrowly focused than a university's CWIS tends to be.

+ Page 34 +

Other administrators are mounting "subject Gophers" that may cover a single subject or a variety of subjects of general interest to users, but not documents pertaining to a particular

campus. For instance, Don Gilbert of Indiana University has deployed a Gopher that is dedicated to materials on biology, and this server is becoming a useful resource for the biological sciences community. Sue Davidsen of the University of Michigan Graduate Library has built a subject Gopher that delivers census, business, and social science data. Known as Go M-Link, this Gopher mainly serves public libraries in Michigan. Subject Gophers in many cases have the most intuitive organization schemes.

The use of the Internet to deliver electronic journals presents a special organizational challenge for the Gopher community. Discussions on how to manage and organize a set of e-journals are underway. Some propose the venerable Dewey Decimal Classification; some suggest the Library of Congress Classification Schedules; others like the Universal Decimal Classification; and others are experimenting with schemes of their own devising. CICNet has announced a project to archive electronic journals via their Gopher. Billy Barron of the University of Texas at Dallas is building this archive, and he is experimenting with organizational issues. [2] His initial effort uses the LC Classification scheme. One group of Gopher administrators and librarians from CICNet, NYSERNet, and other institutions are exploring alternative approaches to Gopher taxonomy.

Some librarians contend that any attempt to fit documents drawn from all areas of human endeavor into a formal classification scheme is not an appropriate model for mounting networked information. Marty Courtois, a librarian and database coordinator at Michigan State University, observes: "Library classification schemes such as the Dewey Decimal System and the Library of Congress classification are good systems for finding materials on the shelf and are necessitated by the fact that a single book can only be placed in one location." [3] With a system like Gopher, cross-references can be set up as simple alternate links. Thus, a set of subject headings based on a controlled vocabulary can make materials far more accessible to the user. As Courtois puts it:

> It's simply easier to browse a classification list to look for "Biological Sciences" than to have to know that "QH300" represents that field. It's like giving the user a chance to look in the card catalog and the shelves at the same time. [4]

The world of networked information is new and evolving. Some organizational schemes that seem obvious to an administrator may be suboptimal for the user. For instance, there is a tendency to categorize networked information servers based on their location or on the underlying technology. For instance, you might find a Gopher menu that offers:

> Other Gopher Servers
> Non-Gopher Campus Wide Information Systems
> WAIS-based information
> World-Wide Web

The user of course is interested in information, not the technology that presents it. In the long run, providers of networked information resources need to find ways to organize

materials by subject matter, not by whether the data resides in a Gopher, WWW, WAIS, or some other technology. As Marie-Christine Mahe of Yale University observes, "A Gopher that splits CWISes along technical lines is equivalent to a supermarket that would shelve tomato sauce in different aisles, according to whether it was in a can or in a glass jar." [5] Mahe believes that when Gopher uses standards such as Z39.58, it will be easier to provide uniform access to data stored under disparate systems. In the meantime, she argues that Gopher administrators should use topical organization whenever possible.

One could imagine a division of labor among Gophers, as follows: (1) publisher Gophers would distribute original material in one or more subject areas; (2) single-subject Gophers would organize and provide access to network resources for a subject area; (3) index Gophers, such as Veronica, would allow users to search for resources by keyword instead of browsing; (4) master Gophers would link users to single-subject and index Gopher servers; and (5) CWIS Gophers would specialize in documents and services that pertain to each campus (they would have links to master Gophers to provide access to Internet resources).

Whether this happens or not, it is clear that there must be more recognition of the specialized roles that Gophers can play. If every Gopher administrator tries to provide both original documents and organized links to Internet resources, these efforts are doomed to fail.