How to build a GopherPupDon GilbertJanuary 1995email: gopherpup@bio.indiana.eduGopherPup is an Internet Gopher+ client with extensions to (a)
display rich text documents, including pictures, and(b)
allow network hypertext links inside documents it displaysThis document includes discussion of i the Gopher+ protocol extentions used between GopherPup and gopher+ serversiihow to create your own gopherpup hyper-rich-text documentsiii Gopher+

server modifications to support GopherPupiv suggestions on how to add hypertext to other gopher+ clientsv GopherPup source code and how to build itvi Current bugs and problems with GopherPupvii why not use Mosaic/WWW instead of extending Gopher+ protocol?I. Gopher+ Protocol Extensions for HypertextThe basic protocol extensions used in GopherPup were outlined in a Usenet article in November 1993 <usenet ref>, available as <gopher+link>. The idea is to

include network hypertext capabilities in the gopher+ protocol as simply as possible. For that, I chose to use the currently defined **gopher-menu** and **gopher+-menu** data types. These data types, when included as a block extension +**MENU**, can be associated with any document served by a gopher+ server. In order to place specific link items within a document, another set of block extensions is needed, which are associated with each item in a +MENU block. These

placing extensions I've called +MENUSTRING, +MENURECT, +MENUBYTES, and others may be included as seems important. Another advantage of this method is that these gopher+menu data can be save to local disk files associated with a document, where users can access them in the same way as from a remote server.Here is the general scheme of a client-server transaction that includes +MENU data. Client connects

to server and queries for a gopher-menu or gopher+-menu. Server responds with requested data. If gopher+-menu data is returned, the server will send any +MENU block extensions associated with any of its documents. If gopher-menu is returned, no +MENU data is returned, and it is the client's responsibility to ask for more information ("!") on an item to see if it has any +MENU extensions.The format that the server sends gopher+-menu data including +MENU blocks is as

follows (the bars indicate tab characters):+-1+INFO: 0MainDoc1 0/Some/Path/0MainDoc1 my.host.edu 70 ++ADMIN: Admin: blah blah+VIEWS:

application/rtf En_US: <12k>

text/plain En_US <8k>

+MENU:+INFO: 0LinkDoc1 0/Some/Path/LinkDoc1 my.host.edu 70 +
+MENUSTRING: “locate at this text in MainDoc1”+INFO: 1LinkMenu1 1/Some/Path/LinkMenu1
my.host.edu 70 + +MENUSTRING: “locate at this other text in MainDoc1”+INFO:
7LinkQuery 7Some/Path/LinkQuery my.host.edu 70 + +MENUSTRING: “locate at this
even different text in MainDoc1” +QUERYSTRING: “search gopher index with this string”+INFO:
1LinkMenu2 1/Some/Path/LinkMenu2 my.host.edu 70 + +MENUSTRING: “locate
somewhere else in MainDoc1”+INFO: 0Doc2 0/Some/Path/Doc2 my.host.edu 70 +
+ADMIN: Admin: blah blah+VIEWS: application/rtf En_US: <12k> text/plain En_US
<8k>+MENU:+INFO: 0LinkDoc1 0/Some/Path/LinkDoc1 my.host.edu 70 +
+MENUSTRING: “locate at etc in MainDoc1”+INFO: 1LinkMenu1 1/Some/Path/LinkMenu1
my.host.edu 70 + +MENUSTRING: “locate at etc in MainDoc1”+INFO: 0LinkDoc2
0/Some/Path/LinkDoc2 my.host.edu 70 ++INFO: 1LinkMenu2 1/Some/Path/LinkMenu2
my.host.edu 70 ++INFO: 0Doc3 0/Some/Path/Doc3 my.host.edu 70 *+[et cetera ...]*The basic idea here is that any document can have an associated gopher+-menu structure. The current U.Minnesota gopherd server can easily include this data by means of its block extensions method. Data in the format of a “.cache” or “.cache+” file can be copied to a a file to be associated with a document (e.g., MainDoc1.gmenu, associated with MainDoc1), or one can create such a .gmenu document by hand. The document maintainer can then edit that file and put in +MENUSTRING and related locating extensions. This will produce a MainDoc1.gmenu file looking like this, for instance:+INFO: 0LinkDoc1 0/Some/Path/LinkDoc1
my.host.edu 70 ++MENUSTRING: “locate at this text in MainDoc1”+INFO: 1LinkMenu1
1/Some/Path/LinkMenu1 my.host.edu 70 ++MENUSTRING: “locate at this other text in
MainDoc1”+INFO: 7LinkQuery 7Some/Path/LinkQuery my.host.edu 70 ++MENUSTRING:
“locate at this even different text in MainDoc1”+QUERYSTRING: “search gopher index with this string”+INFO:
1LinkMenu2 1/Some/Path/LinkMenu2 my.host.edu 70 ++MENUSTRING: “locate at etc in
MainDoc1”The gopherd server should have its configuration file modified to include the block extension type “MENU”, with some suitable filename suffix (e.g., .gmenu). On my server, I have a /usr/local/etc/gopherd.conf that has this entry: blockext: .gmenu MENUThen when gopherd serves out menu information, it will provide it in the above format with any MENU blocks right-shifted by one space.

The +QUERYSTRING is another extension which can be used in conjunction with the +MENU blocks. Its function is to provide a given string that will be used to search a gopher Index database (type 7). That way, a document can easily include hypertext references to data in searchable databases. More about this later.When sent to a client, the client can use the same general method of parsing the +MENU block as it uses to parse standard gopher-menu and gopher+-menu data. Then it parses the +MENUSTRING and related placement extenstions, using that information to place the +INFO link at a particular word, phrase or position in the document. The client software can highlight these hotspots in various ways (underline, color, or in a character-mode client, perhaps as extra symbols in the text like [1]), so the person can see which areas have hypertext links.II. Making Hypertext documents for serving to GopherPupGopherPup now includes the ability to display documents that are in Microsoft standard Rich-Text-Format (RTF). This format was chosen because it is an existing, widely available standard for storing fully formatted documents (including font, paragraph and document styling and pictures). It is suitable for exchange of scientific documents, as it handles symbol fonts, super and subscripting, and the other formatting characteristics needed in most scientific and scholarly documents. Also importantly, the most commonly used wordprocessors on the common computing platforms (Macintosh and MS-DOS/MS-Windows) will read and write RTF format. The primary step in creating network hypertext documents is then as simple as writing the document in your wordprocessor, and saving it in RTF format (see the "Save As" or similar option in your wordprocessor for saving in this format). The next step is to associate network links with words or images in your document. This current release of GopherPup is sparse in its support for associating link data with document parts. Currently this involves (a) creating a separate text file that has gopher+-menu data, (b) inserting +MENUSTRING statements with each +INFO statement. Alternately, if you save a hypertext gopher document retrieved thru gopherpup, it will create a ".go4" gopher+-menu document including the +MENUSTRING statements.II.a) The GopherPup .go4 data file.GopherPup will now look for link data associated with any document it opens from your local disk drives. The standard way to open a local disk file in GopherPup is to choose the File/Open command, and use the system-specific file chooser to select a document to view. When you make such a selection, e.g., MyThesis.rtf or MyThesis.text, the program now will look for an associated file with the suffix ".go4", e.g., MyThesis.go4. The program looks in such a file for gopher+-menu data and if found, will use it to mark hypertext links in the selected document. You can create an initial .go4 data file by saving gopher menu windows from GopherPup (use the File/Save menu when a gopher menu window is the active window). Such a file can be edited with a standard text editor application. It consists of lines such as described above that are the gopher+-menu data:

+INFO: 0LinkDoc1 0/Some/Path/LinkDoc1 my.host.edu 70 ++INFO: 1LinkMenu1 1/Some/Path/LinkMenu1 my.host.edu 70 ++INFO: 7LinkQuery 7Some/Path/LinkQuery my.host.edu 70 ++INFO: 1LinkMenu2 1/Some/Path/LinkMenu2 my.host.edu 70

+Protocols other than Gopher are supported in this format with the addition of +URL: lines. Currently GopherPup requires a +INFO: line for each item, then additional Gopher+ statements will add to and modify that. This is an example:+INFO: 0ItemTitle any/path any.host 0+URL: http://real.host/real/path/hereII.b) Linking with +MENUSTRING statementsCurrently only the +MENUSTRING: “some word or phrase” method of linking a gopher item to a document location is supported in GopherPup. You need to edit the gopher+-menu “.go4” file you created above with a text editor and insert these statements, one following each “+INFO” statement. The syntax of this statement is+MENUSTRING: word+MENUSTRING: “a phrase in double-quotes”+MENUSTRING: ‘a phrase in single quotes’GopherPup will search a document and mark all matches to a given +MENUSTRING as links to the associated +INFO gopher item. II.c) Planned MENU locating statementsFuture versions of gopherpup should expand on these capabilities. It is also hoped that a later version of gopherpup will included means to directly link gopher items to a displayed document, bypassing need to use a text editor. Suggested locating statements are +MENUSTRING: "some string" repeat-value used to locate a network link at a string in a text document. A phrase with embedded spaces should be enclosed in quote (‘) or double quote (“) characters. An optional repeat-value specifies which occurence of the string to match, from the top of the text. The repeat value is not currently implemented in gopherpup -- all instances of the menustring are matched.+MENUSTRING: "bob was here" 3+MENULINE: start-paragraph start-char stop-paragraph stop-char specifies the item's location as a line number and character range in a text document. Start-paragraph indexes a paragraph, from 0 at top, and 'start-char' indexes a character within that line. Likewise Stop-paragraph and stop-char indicate the end of the index. +MENULINE: 100 500 100 515+MENURECT: left top right bottom specifies the item's pixel location rectangle on a graphics document.+MENURECT: 300 0 400 100+MENULINERECT: start-paragraph start-char left top right bottom This is a combinary of +menuline and +menurect, to index a rectangle within a paragraph. +MENULINERECT: 100 500 300 0 400 100+MENUBYTES: startbyte stopbyte specifies the item's location as a byte range in a text document. If 'stopbyte' is missing, only the 'startbyte' location is indexed for the item location.

+MENUBYTES: 100 500+ISMAP: Indicates the item is a queryable graphic map, and x-y coordinates of mouse clicks will be sent to the item server.

II.d) NetDoc formats -- Gopher links combined with document dataAs an alternative to having a separate *.go4* data file of link statements for each document, GopherPup understands a compount document type where the link statements are prepended to the other data. This method allows documents with any kind of data that GopherPup can display, including binary formats such as PICT and GIF, to contain network links. They are typed as NetDoc/Text, NetDoc/RTF, NetDoc/PICT, NetDoc/GIF, et cetera. The format of these NetDoc types is to start with the line+MENU:followed by as many Gopher+ link statements as described above as needed, then the line +DATA:is a key that the document data follows to the end of the file.Here is an example NetDoc header+MENU: +info: 1GopherPup home 1/IUBio-Software+Data/util/gopher/gopherpup ftp.bio.indiana.edu 70 + +menustring: ftp.bio.indiana.edu:/util/gopher/gopherpup +comment: +comment: To read this with a standard wordprocessor, strip out the lines down thru +comment: the "+DATA:" line, so that the first line starts with "{ \ rtf1" +comment: +info: 1DCLAP home 1/IUBio-Software+Data/util/dclap ftp.bio.indiana.edu 70 + +menustring: ftp.bio.indiana.edu:/util/dclap +info: 1IUBio Archive ftp.bio.indiana.edu 70 + +menustring: "IUBio Biology Archive" +info: 7GenBank sequence search 7/.indices/genbank ftp.bio.indiana.edu 70 + +menustring: "fetch a sequence entry from Genbank" +info: 1RTF Info 1/IUBio-Software+Data/util/rtf ftp.bio.indiana.edu 70 + +menustring: "Rich Text Format" +info: 1RTF (Rich Text Format) utilities 1/RTF ftp.primate.wisc.edu 70 + +menustring: ftp.primate.wisc.edu:/pub/RTF +info: 0Go+hypertext-method 0/IUBio-Software+Data/util/gopher/go+menu-article.txt ftp.bio.indiana.edu 70 + +menustring: "network hypertext links" +comment: new gopherpup "mailto" kind 'm' +info: mGopherPup bug report GopherPup@Bio.Indiana.Edu localhost 0 +menustring: GopherPup@Bio.Indiana.Edu +info: 1NCBI Gopher ncbi.nlm.nih.gov 70 +menustring: "National Center for Biotechnology Information" +info: 1ncbi_tools 1/toolbox/ncbi_tools ncbi.nlm.nih.gov 70 +menustring: ncbi.nlm.nih.gov:/toolbox +info: 1Gopher help from U.Minnesota (USA) gopher.micro.umn.edu 70 + +menustring: gopher.micro.umn.edu +comment:+DATA:{\rtf1\mac\deff2 {\fonttbl{\f0\fswiss Chicago;}{\f2\froman New York;}{\f3\fswiss Geneva;}{\f4\fmodern Monaco;}{\f13\fnil Zapf Dingbats;}{\f14\fnil Bookman;}{\f16\fnil Palatino;}{\f18\fnil Zapf Chancery;}{\f19\fnil Souvenir;}{\f20\froman Times;}

*[... and more rtf data...]*III. Gopher+ server modifications to support GopherPupAll of the current extensions to GopherPup will work with the current gopher+ GopherD server from University of Minnesota, I believe. I use one minor extension, but it isn't essential. At this point, I haven't tested other gopher+ servers, such as the Macintosh one. The one extension I've added to server software for these methods, is to allow the combined MENU information and document DATA to be built from two separate files, and sent as one chunk to the client. This extention allows one to support both non-hypertext and hypertext formats from the same data files. For instance, one set of documents at IUBio server is available in the following formats: image/pict, netdoc/pict, image/gif, netdoc/gif, and text/html. The two pict forms use one data file, and the netdoc form is created by prepending a +MENU: block (as in §II.d above). The other three forms are created from one gif file and the same menu block (for netdoc/gif) or with an HTML wrapper (for text/html).IV. Suggestions for adding hypertext support to other gopher+ clientsThe protocol extentions that are outlined here, and embodied in the GopherPup client, are readily transferable to other gopher+ clients. The basic data here is the same gopher+-menu data already parsed by these clients. A gopher+ server should need no modification to send MENU data associated with a document, if it follows the U.Minnesota model of block extensions. It is my intension, and suggestion for others, to provide at least both plain text and Rich Text Format versions of documents for network information servers, since it is usually trivial to produce both forms from a wordprocessor. Thus it isn't necessary for other clients to support RTF display (which is the hard part here -- it took me rougly one day to add just the hypertext portion to my gopher client). Other clients might also care to support other document formats. This method of MENU extensions can be applied to various text and graphics formats.For a client to support MENU hypertext links it basically needs to add recognition of the +MENU: block extention, and extension locator statements like +MENUSTRING:. For a graphics mode client, this would involve highlighting the located strings in a document on display, and responding to user interactions (mouse clicks) with the highlighted item by fetching it from the server, basically in the same way as for gopher items that are displayed and selected from a menu.For character-mode client, it might be easiest to insert markers into a display document to correspond to highlighting in a graphics client. If the document text were Please match this word, and then match this second word, then formats such as Please match this word{a}, and then match this second word{b}, or Please match this word[1], and then match this second word[2], would display easily, and the client could respond to typing of letters or numbers as selection events for the linked items.V. GopherPup source code and how to build itSource code for GopherPup is freely available for non-commercial distribution and use, as are program executables for the major platforms available to the author. It may be modified and enhanced for those who wish to. The source code is available at <ftp://iubio.bio.indiana.edu:/util/dclap/src/> with associated source code in /util/dclap/from-ncbi/, documents in /util/dclap/docs/, and program executables in /util/dclap/apps/.An interesting feature of the GopherPup software is that it is written based on a cross-platform development library. This means that "one-size-fits-all", or at least it tries to. Additions and changes to the program are readily available on the common computing platforms of Macintosh, MS Windows and X/Motif Windows.The cross platform development library includes an ANSI-C interface to Macintosh, MS-Windows, and X/Motif Windows graphic and system environments. The C library is called NCBI Toolkit, with VIBRANT the graphic user interface portion. These parts are written by biocomputing scientists at the National Center for Biotechnology Information

(NCBI). The remainder of the framework for GopherPup is a C++ object oriented, class application library (DCLAP), which is still fairly young. It is based loosely on the Apple Computer MacApp development library (the parent of GopherPup, GopherApp was written to MacApp). VI. Current bugs and undeveloped features of GopherPupGeneral: – RTF and HTML, and in general the rich-text display, need much bug chasing. – HTML Forms are not supported yet (a first hack at it is in progress). – Memory management is improved but still needs much work -- documents are not flushed & loaded as sensibly as they should be. Mac: – seems generally stable, but not completelyXMotif: – somewhat usable, though it still tends to crash. – no printing. This lack is fundamental to the XWindow support of Vibrant/DCLAP.MS-Windows: – freezes fairly frequently still. When not frozen, all the features are there. The 32 bit version seems much more stable -- several of the 16bit version crashes are known due to the small word size of that system for memory allocations, etc. [more bugs are known, just not documented here...]VII. Why not just use Mosaic?Internet Gopher and the network information systems that goes under names of WWW, the Web, HTTP and HTML are both capable of being good network information clients, and in very similar manners. Many of the features that were pioneered in gopher are now being added to WWW/Mosaic client-servers. Many of the same problems are now being addressed. My opinion is that Gopher has good features that should not be lost sight of due to a desire for hypertext, and a pretty (inter)face.The hypertext markup language (HTML) has advantages and disadvantages over the method discussed here. One basic difference is that HTML is an in-line markup language, where hypertext links are inserted in the document. The form described here doesn't change documents themselves, but adds a secondary data file, or is appended at the start of the data. In some cases this nicer, as it leaves the original document in its original form, without confusing the text with network link codes. GopherPup has been developed to allow translators to display various document formats, and as well to permit adding network links to any of these formats, whether they are simple text or some binary image format.

The HTML document format as implemented in Mosaic and other WWW browsers does not provide enough support for the formatting features needed for scientific documents. Symbol and other fonts, super and subscripts, various formatting capabilities that any common wordprocessor can display are essential for electronic publication of scientific documents. Also, to get wordprocessing documents into HTML format involves various difficulties. The RTF format chosen for GopherPup is better suited to display of scholarly articles, and has the great benefit of being readily produced by many or most of the commonly used wordprocessing programs. As well, GopherPup has the markup features needed to allow you to very simply add network links to any part of a displayed document, by selecting words or portions of the display and dragging a network link item to that selection.The RTF support also includes support of embedded pictures, including both vector and bitmap forms. Vector pictures are not supported in Mosaic/www, but are very useful for scientific data (frequently graphs) both because they are independent of resolution of the display/output device, and also because they can be re-edited to different sizes or components can be changed easily. In one example of genetic map drawings, I have compared the same maps in formats suitable for Mosaic browsers (GIF) and for GopherPup (PICT). The low-weight, high-resolution vector map wins hands down any speed contests, at about 5 times faster in fetching and displaying the same map, making it much more suitable to set up interactive graphic browsing services with GopherPup. The hypertext transport protocol (HTTP) is essentially the same in functionality to the Internet Gopher protocol, it just works with different codes. Since gopher servers can also server HTML documents if those are preferred, there seems no compelling reason to run an HTTP server. Gopher already has mature support for client-server handling of multiple-format documents (Gopher+ VIEWS). This allows for instance a server to provide both a fancy, picture-rich document and a simple, plain text one that can provide the same basic information. But this choice of formats lets gopher serve a wider range of client and network transport capabilities.Mosaic and other so-called web browsers tend to be “kitchen-sink” clients. They try to speak many network protocols, but are masters only of HTTP/HTML. In part this means that people using these browsers get a poor impression of methods other than HTTP/HTML, because the browser only poorly support other methods. This is basically why I avoid using the terms “web” and “WWW”. The advocates of HTML/HTTP browsers are claiming more than they deliver. Gopher clients tend to concentrate on mastering one protocol, and rely on information servers to provide gateways to alternate protocols if desired. The HTML/HTTP developers have long talked about the “standard” that they developed. It currently looks like it will be very difficult for anyone to extend or improve that method, as they have a committee in charge of it. In my estimation, standards are not born over night. Where standard make sense, existing ones should be employed before inventing a new one. Where new methods are needed, they should be developed and tested before calling them a standard. That is why I’ve (a) concentrated on using existing formats like RTF and PICT that are already in wide use for document display, and (b) extended and experimented with the flexible Gopher+ protocol to get it to do useful things, rather than try to do such with the restricted HTTP/HTML protocols.Both Internet Gopher and WWW are still evolving and changing rapidly. It may well be that they converge on some set of features that all find useful. GopherPup is one step to

providing a complex client that supports the important information protocols. I've hoped with GopherPup extentions to amend the parts of Gopher that many people find advantageous in Mosaic, and perhaps provide in some small ways a better network client for some uses.The evolving GopherPup now supports local gophering -- browsing and (soon) searching of local data with the same simple methods that apply to remote services. With this addition it will be feasible for information providers to offer both network services and CDROM based information disks relying on one set of data and client/user-interface.