Journal of Universal Computer Science, vol. 1, no. 4 (1995), 235-246
submitted: 3/8/94, accepted: 12/10/94, appeared: 28/4/95, ©Springer Pub. Co.

Evolution of Internet Gopher

Mark P. McCahill
(University of Minnesota, USA
mpm@boombox.micro.umn.edu)

Farhad X. Anklesaria
(University of Minnesota, USA
fxa@boombox.micro.umn.edu)

Abstract: How the Internet Gopher system has evolved since its first released in 1991 and how Internet Gopher relates to other popular Internet information systems. Current problems and future directions for the Internet Gopher system.

Keywords: Distributed Information Systems, Internet Gopher, Gopher+
Category: H.5.1

1 Introduction

This paper considers how the Internet Gopher system has developed since its initial release in the spring of 1991, and some of the problems that are driving further evolution of Gopher and other popular Internet information systems. Although two of the most popular new Internet information systems (Gopher [McCahill (1992)] and World Wide Web [Berners-Lee (1992)]) have become quite widely used, they both have some basic architectural deficiencies that have become apparent as the systems were deployed on a large scale. These deficiencies limit the long term stability of the information stored in these systems. For instance, both Gopher and World Wide Web (WWW) refer to information by location, and so both systems are plagued by references to information that either is stale, has moved, or no longer exists. Beyond the architectural problems, the volume of information being published using Gopher and WWW requires continued evolution of user interfaces and categorization/searching technologies. The imminent arrival of more Internet-aware page description languages must also be accommodated by these systems, and will have a significant impact on how client applications are written.

2 The Original Internet Gopher

Internet Gopher was originally designed to be a simple campus wide information system [Lindner (1994)] to address local needs at one institution (the University of Minnesota). The design philosophy of Gopher was to make it possible for departments and other groups at the University of Minnesota to publish information on their own desktop systems and to hide the distributed nature of the servers from the user using the system [Alberti (1992)].

2.1 A User's View of Gopher

From the user's perspective, Gopher combines browsing a hierarchy of menus, viewing documents selected from menus, and submitting queries to full-text search engines (which return menus of matches for the queries). The structure of the menu hierarchy that the user sees depends on which Gopher server the user first contacts, so there is no notion of central control of the Gopher menu hierarchy. Gopher server administrators run their systems autonomously and are free to organize both the information on their server and references to information on other servers to meet their needs. This architecture makes it possible for the Gopher system to scale up well since there is no central (or "top level") server to get overloaded. On the other hand, because servers are autonomous, it is the responsibility of each server administrator to make sure that references to items on either server are not pointing to stale or nonexistent information. Unfortunately, not all server administrators are vigilant in maintaining the quality of their links to other servers. This is analogous to the problem in WWW of documents containing dangling and stale URLs.

Because of Gopher's distributed architecture, there are many different organizations of the information in the Gopher information space (Gopherspace). It is expected that users will naturally prefer to start their Gopher clients by pointing them to their favorite or home Gopher servers. This distributed architecture encourages the formation of communities of interest which grow up around well-run Gopher servers. If there is no central server and users are expected to naturally gravitate to favorite home servers how does the client software know where to start?

Each user's Gopher client software can be configured with the address and port number of the first gopher server to contact when the client software is launched. When the client software contacts this server, the server returns a list of items (a menu) to the client. Items described in the menu either reside on the server or are references to items residing on other servers. These references to items on other Gopher servers tell the Gopher client the domain name of the server, the port number of the Gopher process, the type of the item (document, directory, search engine, etc.), and the Gopher selector string to be used to retrieve the item. Given the minimal information required to describe the location of an item on a Gopher server, it is easy for server administrators to add references to other servers ("links"), and many servers main value is as subject-matter specific collections of references. Gopher client software also generally provides a facility for saving references ("bookmarks") to items selected by the user.

While the references to items on other servers would seem to be Gopher-specific, this is not strictly true. It is possible to map enough information into a Gopher descriptor to make it possible to describe how to access most popular Internet services, and Gopher descriptors are used to reference items on HTTP servers, finger servers, telnet sessions, CSO/Ph phone books, and via gateways X.500, WAIS, Archie, and anonymous ftp servers.

2.2 Growth of the Gopher System

Although originally designed as a campus wide information system, Gopher was rapidly adopted by a variety of institutions and as of November 1994 it is estimated that there are over 8000 servers on the Internet. Part of the reason for Gopher's widespread adoption as a platform for publishing on the Internet is the ease of setting up and running a server. Server administrators typically run a gopher process and publish a part of their local file system as a gopher hierarchy. By default, the names of the items in the Gopher hierarchy are the filenames of the documents and directories published by the server. Because Gopher accommodates a variety of page description languages there is no need to reformat or markup documents to be published.

Another reason for Gopher's popularity is that server administrators can add links to other Gopher servers and give their users access other collections of information. Since users are not necessarily aware where the information they are accessing resides, server administrators can easily take advantage of each other's efforts.

2.3 Locating Information with Gopher

Part of Gopher's popularity is due to the explicit organization of information in the Gopher system which lends itself to quickly finding information. While hypertext documents are certainly useful as one type of information content, it is not clear that a system consisting of nothing but hypertext documents solves all the problems in an information system. In fact, it is difficult to quickly make sense of a complex, visually rich document. If one goal is to allow users to quickly navigate, it is clear why browsers for file systems always have provisions for a menu-like listing of items: a predictable, consistent user interface (such as a menu) is easy to make sense of quickly, and this is necessary for rapidly traversing an information space. Menu-based system also have advantages in that they can be mapped into a variety of user interface metaphors and browsers and can be compactly represented. Since one of the design points for the Gopher system is to run quickly even over low bandwidth links, a compact representation of a collection (menu) allows users with slow machines to quickly traverse Gopherspace to locate documents of interest.

A good metaphor for current information systems on the Internet to compare them to a book. Books generally are structured into three functional sections: a table of contents, an index, and the pages in the middle of the book (the content). Using this metaphor, Gopher acts something like a book's table of contents since Gopher presents a structured overview of the information and makes it possible to jump directly to the section for a specific topic. Search engines (such as WAIS, Archie, and VERONICA) are similar to the index in the back of the book because they allow users to jump into the content based on one or more words of interest. To complete the metaphor, the pages in the middle of the book are represented in a variety of formats on the Internet; content may be in the form of text, graphics, or page description and markup languages such as postscript, PDF, and HTML. The Gopher protocol can of course provide access to search engines and to document content in different formats.

When an explicit hierarchical organization of information is combined with a searchable index of titles of objects in Gopher (such as the VERONICA index of all Gopher items or a Jughead searchable index of items on a single server) users have the choice of either browsing menus or submitting queries to locate items. The scope of the queries can be either global (VERONICA) or local to a specific server (Jughead), and users can select the scope of their search by traversing the hierarchy to locate an appropriate search engine. Gopher has always been designed to be a hierarchical framework to organize collections of documents and search engines rather than a monomorphic Internet-aware page description language.

2.4 Architectural Limitations

The ease of Gopher server setup goes hand-in-hand with an architectural limitation of Gopher. To run a Gopher server does not require either setting up any sort of replication system or formal agreement with other sites to replicate or mirror information. The lack of a formal system for propagating redundant copies of information stored on Gopher servers means that users may be going to the other side of the world (or across a congested network) to fetch a copy of a document that is also stored locally. Moreover, popular servers are heavily loaded and there is no systematic way of spreading the load to other sites (although ad hoc schemes such as informal mirroring agreements are used to partially address this problem). Note that this architectural limitation is also inherent in WWW. Referring to items on other server by location rather than by name makes it easy to add new servers since there is no registration process required as items are added to (or removed from) servers.

Referring to items by location rather than by name also means that there is no name to location mapping service to be maintained, and clients do not have to go through a name to location resolution before attempting to fetch an item. Unless the name to location mapping service is extremely fast, it has the potential for being a significant performance bottleneck. Given the scale of the Internet, it is clear that any name resolution service will have to be replicated and distributed to scale up properly, but this means that the client will have to first locate the appropriate name mapping server (potentially a slow process) before it can proceed with a name to location lookup.

Clearly, there are significant engineering tradeoffs between a system based on name-to-location resolution and a system that only refers to information by absolute location. The direction the Gopher system is taking to address these architectural problems is to accommodate both reference by location and reference by name. The expectation is that clients will first attempt to resolve the reference by location and if that fails (or the client finds that the server responds slowly), the client will attempt a name to location lookup to find other locations that hold the item of interest. To accommodate multiple references to items by both Uniform Resource Locators (URLs) and Uniform Resource Names (URNs) requires a place to store the references as some sort of meta-information. The original Gopher protocol had no provisions for meta-information, but as we will see later in this paper, Gopher+ solves this problem.

2.5 Gopher Gateways to Other Information Systems

Soon after the initial release of Gopher, a concerted effort was made to develop software gateways to give Gopher clients access to information on servers such as anonymous ftp, X.500, NNTP, and WAIS. These software gateways made information on other systems visible to Gopher users and handling the translation between Gopher requests and the protocol on the target system (for instance: ftp).

The decision to write software gateways was driven by the desire to keep Gopher clients small and simple. Rather than building support for several protocols into the Gopher clients, clients that only understand Gopher protocol can be small, simple, easily written, and run on personal computers that are relatively slow and have limited memory. Gateways for Gopher clients greatly expanded the information Gopher users could access. Not surprisingly, the gateways became quite popular. Unfortunately, excessively popular machines on the Internet tend to be either slow or the machines must throttle back the demand for their services by refusing requests when they are busy.

Since the original Gopher protocol had no provision for expressing meta-information about an item, there was no place in the protocol to tell clever clients which might speak several protocols that the an item could be fetched directly (for instance via an ftp session), so even clients that had the capability for go direct to a non-Gopher service such as ftp had no choice but to go through a Gopher software gateway. Clearly, references to information available via a different protocol ought to enumerate the possible paths to the information (both through the gateway and directly via the other protocol). To take advantage of the work being done to develop URL-aware clients, the enumerated references should be made as URLs. Not surprisingly, the architectural problems and the lack of meta-information in the original Gopher protocol are addressed by the Gopher+ protocol.

3 Gopher+ (the Second Generation Gopher Protocol)

After the first year of Gopher deployment, it had become obvious that Internet Gopher was going to be more widely used than the designers had originally anticipated and that there was a need for an enhanced protocol which retained backward compatibility with the original Gopher protocol. Essentially it was necessary to move beyond a distributed menu system to become a true information system by treating Gopher menu items as objects with extensible collections of attributes (meta-information) associated with each object. A set of upward compatible extensions called Gopher+ were added to the original protocol [Anklesaria (1993)].

3.1 Maintaining Backward Compatibility

To maintain compatibility with the installed base of Gopher servers required finding a way for the new servers to announce their capabilities to Gopher+ clients without breaking old clients. Since old clients parse the item descriptors returned by Gopher servers by looking at the information returned as fields separated by the tab character, it was possible to add another field at the end of the item list. To announce their ability to understand Gopher+ verbs, Gopher+ servers return an extra field (most often a "+") along with Gopher item descriptions. Non-Gopher+ clients ignore the field, while Gopher+ clients use the field to recognize Gopher+ servers.

3.2 Gopher+ Item Attributes

Once a Gopher+ client recognizes a Gopher+ server, it can take advantage of features of the Gopher+ protocol to fetch meta-information about items on the server. The Gopher+ protocol includes verbs that allow clients to request specific types of attributes by name or request all attributes for a Gopher document or directory. For instance, a client may request only the abstract for an item, or it may request all meta-information about an item.

Figure 1 shows a typical Gopher+ item attributes for a document that is available in several languages. The item attributes are defined in labeled blocks. The +INFO block contains the Gopher item descriptor, the +ADMIN block contains information about the owner of the item and the modification date, and the +VIEWS block enumerates the alternate views of the item. Gopher+ alternate views are used to make items available in several data formats (such as text, postscript, PDF, etc.) and describe the language the document in which the document is written. Gopher+ clients typically fetch the alternate views and give the user a choice in which view of the document is to be displayed. Gopher+ clients have built-in support for popular data formats and when necessary can call external helper applications to render data formats not supported inside the Gopher+ client.

```
+INFO:0Welcome 0/Welcome bogus.micro.umn.edu 70 +
+ADMIN:
 Admin: Joe Blow +1 612 625 1300 <joe@bogus.umn.edu>
 Mod-Date: Tue Feb  9 17:06:56 1993 <19930209170656>
+VIEWS:
 Text/plain En_US: <.1k>
 Text/plain De_DE: <.7k>
 Text/plain Es_ES: <1k>
 Text/plain Fr_FR: <.9k>
+ABSTRACT:
 This file tests out the various languages in gopher+
 You should be able to choose your favorite language.
```

Figure 1: Gopher+ attributes for a document available in several languages

Figure 2 shows the item attributes for a Gopher directory. In this example there is a Time To Live (TTL) value for the directory available as an advisory to clients about how long (in seconds) to cache the directory contents. Note that there are alternate views of the directory; this server can provide Gopher, Gopher+ and HTML versions of the directory.

```
+INFO: 1A_Directory 1/dir bogus.umn.edu 70 +
+ADMIN:
 Admin: Joe Blow +1 612 625 1300 <joe@bogus.umn.edu>
 Mod-Date: 12/15/94 12:54:09 <19941215125409>
 TTL: 1800
+VIEWS:
 application/gopher-menu En_US: <1k>
 application/gopher+-menu En_US: <2k>
 text/html En_US: <10k>
```

Figure 2: Gopher+ attributes for a directory

3.3 Item Attributes and Extending Gopher's Capabilities

In addition to the Gopher+ attributes already discussed, there are attributes to hold keywords, the definitions of electronic forms to be displayed and filled out by the user, and authentication/encryption methods to be used for access to the item. Gopher+ item attributes provide an open-ended method for extending Gopher's capabilities, and as schemes for referring to items by name (rather than location) are deployed, the item's name can be stored in a Gopher+ item attribute.

Gopher+ item attributes also provide an obvious container to store references to redundant copies of an item whether the reference is by location (such as a URL) or by name (such as a URN). As discussed later in the paper, we are currently working on extending the Gopher user interface metaphor to encompass 3D scenes. By defining a new Gopher+ attribute block, Gopher is easily extended so that it can hold definitions of 3D polyhedra, textures to be mapped onto the surface of the 3D object, and the location of the object in a scene.

Unlike WWW, Gopher keeps information hierarchy structure and meta-information separate from document contents. Current WWW implementations store all information inside documents. The approach of embedding everything inside the document is much less flexible than storing meta-information separately. Separately stored meta-information makes it is possible to find out how large an item is before starting to fetch it, and to know what sort of authentication methods are required for access before attempting access. Moreover, a server administrator that is referring to a document stored on another server can write and store meta-information without changing the contents of the document. This is important for administrators that maintain references to information published outside the control of the server administrator.

4 Future Directions for Gopher and Related Systems

In the three and a half years since the first version of Gopher was deployed, there have been significant changes in the Internet and computing landscape. The computational capabilities of desktop systems have improved significantly and affordable desktop PC systems that can render simple 3D scenes are now starting to reach the market. In addition, support for component documents is starting to become part of operating systems such as Windows, the Macintosh OS, and Unix as the OpenDoc and OLE document architectures are refined and implemented. The acceptance of Gopher and WWW and the growth of the Internet have demonstrated the utility of distributed information systems designed for use by non-technical users. Where are all these developments leading us?

4.1 More Internet-Aware Page Description Languages

One near-term direction is the development of more widely deployed Internet-aware page description languages. The success of HTML as a vehicle for advertising and marketing is prompting vendors of page description languages to incorporate Internet references into their documents.

For instance, by embedding URLs inside Adobe Acrobat PDF files, it is possible to give PDF documents similar capabilities to HTML documents. A freely available extension named InternetLink (available as part of the TurboGopher version 2.0 release [Anklesaria (1994)]) for use with the Adobe Acrobat Exchange program was recently released by the Gopher software development team at the University of Minnesota. InternetLink allows users to embed URLs into PDF documents by defining hypertext links with the Adobe Acrobat Exchange software. When a user clicks on a link, the InternetLink software looks at the scheme of the URL and calls the appropriate client (Gopher, WWW, NNTP, ftp, etc.) to resolve the URL. Since Internet client writers are now releasing clients that accept "get this URL" messages from other applications, it is not necessary for the software that renders pages to know about Internet protocols. All that is required is the ability to call other applications with a "get this URL" message. Adopting this technique for use in page description languages and component document architectures is straightforward and should become common over the next year.

While some argue that HTML has unstoppable market momentum, HTML is not without problems. The page markup orientation of HTML seriously limits the control the author has over presentation. Unlike HTML, PDF is a page description language, so the author of a PDF document has complete control over the layout, fonts, and look and feel of the document and this is considered to be a key feature by magazine and newspaper publishers and graphic designers.

We expect a proliferation of page description languages and documents that incorporate hypertext links to Internet resources. It is unlikely that HTML will be the only sort of widely deployed Internet-aware document type. Accommodating the proliferation of page languages is easy in the Gopher architecture since Gopher does not try to represent all information in on page markup language; Gopher+ was specifically designed to support a variety of document formats.

4.2 Component Network Applications

With the proliferation of Internet-aware document types, it will be increasingly difficult to write a single application that can render all possible document types. In the same vein, the proliferation of Internet protocols makes it difficult to write Internet clients that fully support a wide range of protocols. While monolithic applications such as Mosaic [Andreesen (1993)] claim to support all popular Internet protocols, close examination of these all-in-one applications reveal that they support subsets of the protocols and require excessive system resources on the user's machine. Given the continued evolution of Internet protocols it is our belief that monolithic applications will always deliver poorer performance than dedicated ftp, WWW, NNTP, WAIS, or Gopher clients. As more sorts of applications (for instance: video conferencing) become popular it will be increasingly impractical to write monolithic applications. Large software developers such as the OpenDoc consortium (Apple

Computer, Word Perfect, Novell, IBM, and others) and Microsoft (OLE) are in the process of building component frameworks, and Internet applications will benefit from these frameworks since they will give users the appearance of an integrated application without requiring development of monolithic applications.

Given standards for representing Internet resource locations (URLs) and names (URNs) it is attractive to write single purpose components that call each other by passing URLs or URNs. This sort of architecture makes it easier to update components as protocols evolve and keeps the Internet tool business from being dominated by a few large code factories that develop and maintain massive monolithic do-it-all applications. This component architecture also works well with developing more Internet-aware document types, since the document rendering application can call the appropriate client application to resolve URLs.

4.3 New User Interface Metaphors

The user interface used in today's Gopher is that of traversing a hierarchy of menus and search engines. WWW's user interface metaphor is to treat everything as a hypertext document. While both these user interface metaphors have obvious strengths, they are also both an artifact of the computational limitations of the systems widely available three or four years ago. Given significantly faster systems available today, we believe that using 3D scenes as a user interface metaphor is worth exploring [McCahill (1994)]. Both Gopher and WWW users complain of feeling "lost in cyberspace" and this problem may be addressed by a user interface that is more similar to navigating through the real world.

Beyond trying to address the "lost in cyberspace" problem, a 3D user interface will give us a more powerful way of representing relationships between documents and the results of searches. The growth in the number of documents available in systems such as Gopher and WWW creates the problem of being able to find and categorize very large collections of documents. Indexes of documents searchable by name (as in Archie or VERONICA) are valuable, but we are rapidly reaching the point where there are so many documents that nearly all queries return a very large number of matches. The same problem can be seen with full-text indexes of large collections of documents; nearly any query returns a very large number of documents. A 3D interface provides more degrees of freedom for representing relative "closeness" of documents than a menu or a document can.

While 3D scenes as a user interface should provide a valuable adjunct to the current user interface metaphors, there is the question of how to apply this interface to existing servers. It is unclear how this could be accomplished for WWW servers since there is no clear structure to the information space other than that embedded inside the documents. Since Gopher keeps the structure of the information space external to the documents it is possible for a 3D client to automatically generate scenes from current gopher directories. For instances, the items in a Gopher directory can be represented as objects in a scene and their names can be written onto the face of the objects. Since Gopher+ supports meta-information, it is also straightforward to add information to override default client-generated scenes for a menu. Development of these sorts of 3D scene generating clients will help drive development of better document classification and clustering schemes by making it easier to visualize the relationships between documents.

5 Concluding Remarks

Gopher has evolved from a simple campus wide information system to a popular, widely deployed information system that makes the Internet appear to be a single large hierarchical collection of documents and search engines. The most serious architectural limitations of the original Gopher protocol were addressed by the Gopher+ extensions to the protocol and we believe that Gopher is well positioned to accommodate a proliferation of Internet-aware document types and new user interface metaphors made possible by faster desktop computers.

6 Acknowledgements

Grateful acknowledgment is made to Hermann Maurer, Frank Kappe, and Kieth Andrews of the Institute for Information Processing and Computer Supported New Media (IICM) at the Graz University of Technology for many enlightening discussions of distributed information systems on the Internet. We also wish to thank Shih-Pau Yen at the University of Minnesota for his encouragement and support of the Gopher development team.

References

[Alberti (1992)] Alberti, B., Anklesaria, F., Lindner, P., McCahill, M., Torrey, D.: "The Internet Gopher Protocol: a distributed document search and retrieval protocol."; University of Minnesota. available by anonymous ftp from boombox.micro.umn.edu in the directory /pub/gopher/gopher_protocol/ <URL:ftp://boombox.micro.umn.edu/pub/gopher/gopher_protocol/>

[Andreesen (1993)] Andreesen, M.: NCSA Mosaic technical summary. Available by anonymous ftp from ftp.ncsa.uiuc.edu as file /Web/mosaic-papers/mosaic.ps.Z <URL:ftp://ftp.ncsa.uiuc.edu/Web/mosaic-papers/mosaic.ps.Z>

[Anklesaria (1993)] Anklesaria, F., Lindner, P., McCahill, M., Torrey, D., Johnson, D., Alberti, B.: "Gopher+: upward compatible enhancements to the

Gopher protocol."; University of Minnesota. Available by anonymous ftp from boombox.micro.umn.edu in the directory /pub/gopher/gopher_protocol/Gopher+ <URL:ftp://boombox.micro.umn.edu/pub/gopher/gopher_protocol/Gopher+/>

[Anklesaria (1994)] Anklesaria, F., Johnson, D., et.al..: University of Minnesota. TurboGopher software version 2.0. Available by anonymous ftp from boombox.micro.umn.edu in the directory /pub/gopher/Macintosh-TurboGopher/ <URL:ftp://boombox.micro.umn.edu/pub/gopher/Macintosh-TurboGopher/>

[Berners-Lee (1992)] Berners-Lee, T., Cailliau, R., Groff, J., Pollermann, B.: "World Wide Web: the information universe"; Electronic Networking: Research, Applications and Policy, 2, 1 (1992) 52-58.

[Lindner (1994)] Lindner, P.: "Internet Gopher user's guide"; University of Minnesota. Available by anonymous ftp from boombox.micro.umn.edu in the directory /pub/gopher/docs/ <URL:ftp://boombox.micro.umn.edu/pub/gopher/docs/Papers/>

[McCahill (1992)] McCahill, M.: "The Internet Gopher: A Distributed Server Information System"; Connexions - The Interoperability Report, 6, 7 (1992) 10-14.

[McCahill (1994)] McCahill, M., Erickson, T.: "A Preliminary Design for a 3-D Spatial User Interface for Internet Gopher"; University of Minnesota. Available by anonymous ftp from boombox.micro.umn.edu in the directory /pub/gopher/Gopher_Conference_94/Papers/ <URL:ftp://boombox.micro.umn.edu/pub/gopher/Gopher_Conference_94/Papers/>