Hyper-G: A New Tool for Distributed Hypermedia

Frank Kappe, Keith Andrews, Jörg Faschingbauer,
Mansuet Gaisbauer, Michael Pichler, Jürgen Schipflinger

Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology,
Graz, Austria

Abstract

This paper describes Hyper-G, a new hypermedia information system which combines the best of Gopher, WAIS, and World Wide Web. Hyper-G is specifically designed as a distributed, large-scale hypermedia information system supporting navigation in a large body of dynamically changing information without becoming "lost in hyperspace". Users may choose a hierarchical navigation paradigm, click on hyper-links, go on guided tours, or perform variable-scope searches.

This paper presents Hyper-G from the user's perspective, outlines the basic architecture of the system, and describes its interaction with existing distributed information retrieval tools like Gopher, WAIS, and World Wide Web.

1 Introduction

Hyper-G is the name of an ambitious hypermedia project currently in progress at Graz University of Technology. Hyper-G was designed as a general-purpose, large-scale, distributed, multi-user, hypermedia information system, similar in scope to Xanadu [13] and Intermedia [7]. Based on previous experience with large-scale information systems (videotex), the aim of the Hyper-G project is to develop a flexible hypermedia framework in order to study and as far as possible eliminate the problems typically associated with large-scale (i.e. containing millions of documents) hypermedia systems:

1. **Disorientation:** Also known as the "lost in hyperspace" syndrome, this phenomenon has often been addressed in the literature (first by [5]). The symptoms are: the difficulty of gaining an overview, finding information again, not knowing how much information there is on a subject and how much of it has already been seen, not knowing whether everything important has been seen, etc. For small-scale systems (like typical HyperCard or ToolBook applications), the problem is not so acute. However, for large-scale, dynamic hyperbases special facilities have to be provided [18, 6]. The approach taken in Hyper-G is described in Section 2 of this paper.

2. **Authoring:** Large hyper-structures cannot be created by a single person. As in large-scale software development, it would be desirable to rely on reusable hypertext modules with well-defined interfaces which are created by different authors according to a set of predefined rules, rather than the current practice, where the simple node-link model of hypertext facilitates the creation of "spaghetti links" (the analogy of the node-link model with the GOTO statement of programming languages has been discussed in [4]). This problem is not discussed further in this paper, however, the interested reader is referred to [12, 17].
3. **Information Distribution:** It is clear that a truly large-scale hypermedia system will, by necessity, also be a distributed information system. Hyper-G is based on the client-server model, with clients and servers connected via the Internet – Section 3 describes the architecture of Hyper-G. Furthermore, it is desirable to provide as much interoperability as possible with similar existing tools and services, like Gopher [1], WAIS [16], and World Wide Web (WWW) [2]. The interoperability of Hyper-G is described in Section 4 of the paper.

2 Browsing and Searching in Hyper-G

In order to offer the user a convenient way of finding information in a large-scale hypermedia information system, a combination of sophisticated navigation and retrieval facilities is necessary. We believe that a blend of hyper-navigation, hierarchical structuring of information, guided tours through the information universe, and various search facilities in conjunction with state-of-the-art user interface metaphors will serve the user better than any single technique alone.

It should be noted that the following discussion deals only with the *logical* structure of information. The *physical* structure (i.e. which parts of the information body are stored in what part of the distributed system) should be seen as completely independent of the logical structure.

2.1 Hyper-Navigation

In a hypertext system the user navigates by clicking on a "hot spot" (a so-called *source anchor*) attached to a *document*, which in turn activates a *link* to a certain area in another document (the *destination anchor*) or the whole *destination document*. The new document is then visualised, the user may again click on anchors, and so on.

This simple and intuitive paradigm of browsing through information rather than searching for it has already successfully been used in a number of information systems (for example WWW), and it is indeed very well suited for applications such as encyclopedias, user manuals, computer based teaching material, presentations, online help systems, and the like. The use of multimedia document types (image, sound, video, animation) in addition to text transforms hypertext into hypermedia. In hypermedia systems the browsing paradigm is essential, as multimedia documents cannot in general be searched for. This is why Hyper-G was designed as a hypermedia system.

However, as explained in the introduction, the basic node-link model of hypertext has its limitations when applied to a large body of information, both for the user ("lost in hyperspace") and for the author ("spaghetti links"). Therefore, this basic model is augmented in Hyper-G by additional structuring and navigation facilities – *collections*, *guided tours*, and *searching* – which are shown in Figure 1 and discussed in more detail in the forthcoming sections.

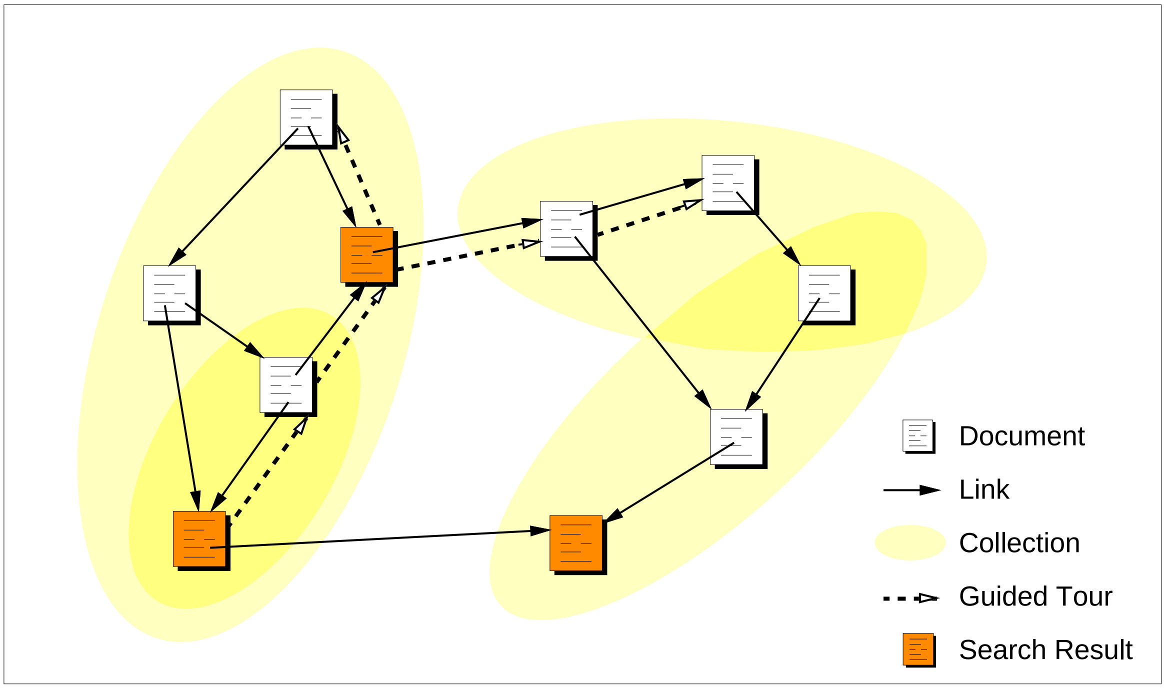

Figure 1: The Hyper-G Navigation Model

2.2 Collection Hierarchy

Every Hyper-G document is a member of one or more *collections*, which are in turn members of one or more collections (except for the *root collections*). This (recursive) definition yields *collection hierarchies* like the one shown in Figure 2. Note that this hierarchy is, in fact, an acyclic directed graph and not a tree, since objects may belong to multiple "parent collections". Rather, the collection hierarchy can be seen as a thesaurus-like structure defined over the whole information body and orthogonal to the hyperlink structure (compare Figure 1).

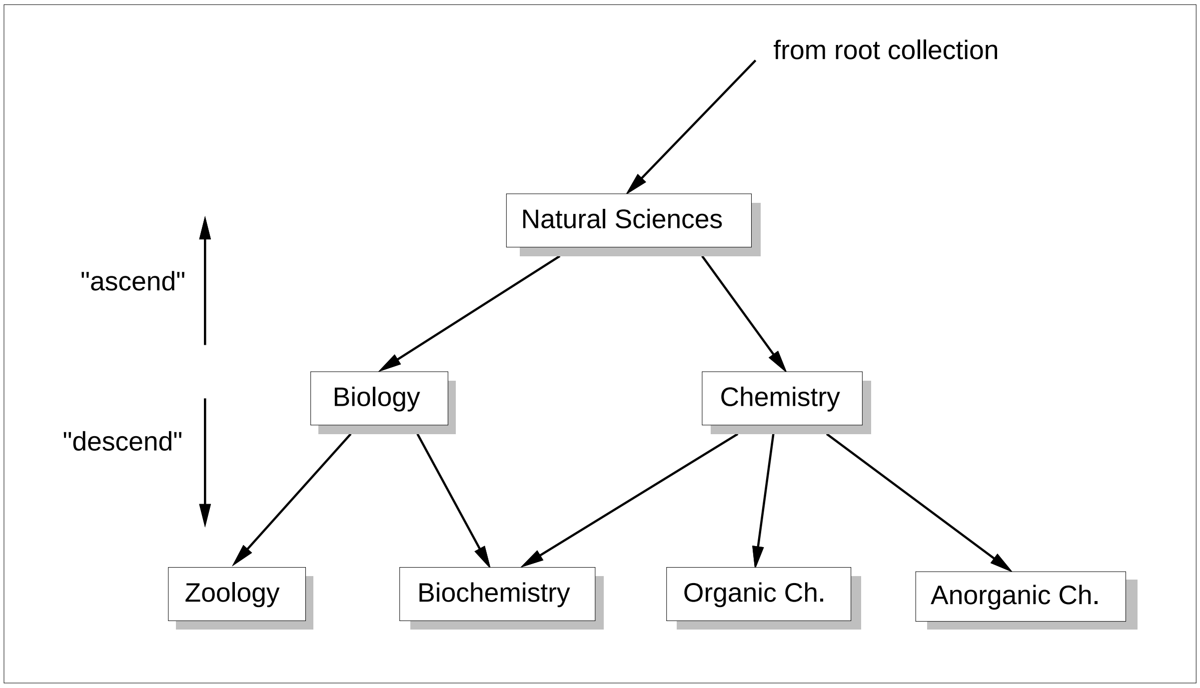

Figure 2: The Collection Hierarchy

The collection hierarchy serves three purposes:

1. **Navigation**: When users visit a collection, they are given an overview of all objects belonging to that collection, and can select (and thus visit) any of these objects (similar to a Gopher menu). This hierarchical navigation scheme reduces the propability that users get lost in the information space. In addition and in parallel to this paradigm, Hyper-G users may use hyper-navigation, guided tours and searching to visit documents. In this case, they may consult the collection hierarchy to learn the relative position of that document within the information universe (see Section 2.5), which alleviates the "lost in hyperspace" syndrome.
2. **Search Scope**: When issuing a search operation (see section 2.4), users may indicate that only parts of the collection hierarchy should be searched, in order to reduce the amount of matches returned (for example, by switching off dictionaries when one knows the meaning of the search term). Users may mark certain collections as "active": only objects which are members of the active collections (recursively) are returned by a search operation.
3. **Access Rights**: In a large-scale hypermedia information system we assume there exist many authors. The collection hierarchy is used to grant write permissions to certain parts of the information structure to authors, and to organise registration of their information with the rest of the information universe. Also, read access to certain parts of the information space can be granted or denied to certain user groups.

In our concrete implementation, three classes of collection are currently defined:

- Ordinary *collections* as described above display a list of menu items when visited. The order of the items may be statically defined or dynamically sorted based on certain attributes of the collection members (title, author, creation time, etc.).
- A *cluster* is similar to a collection, but when a cluster is visited all of its substructures are visited (visualised) too. This structure is used to implement multimedia documents (for example, to play an audio clip while an image and a text document are shown), and to support multilingual documents and version control.
- A *tour* is a collection, which when visited visits all of its substructures in a certain order. It is used to implement "linear hypertexts" with automatically generated "next" and "previous" interface elements. This class of collection is used to realise *guided tours* (see Section 2.3).

The Hyper-G database server is aware of these structural elements and uses them to maintain database consistency (such as making sure that every document is a member of at least one collection, deletion of a document from a tour automatically joins its neighbours, etc.)

2.3 Guided Tours

Guided tours are paths through the information network which have been signposted by some expert in the topic at issue. As shown in Figure 1, guided tours need not take into account the link structure or collection membership of the visited documents.

Of course, a document may be part of any number of such tours. When users visit a document as part of a tour, they are given the opportunity to switch to a different tour (if available), or continue with a different navigation paradigm. Guided tours are typically used to present a compilation of highlights about a particular topic, very much like taking a guided tour to see the sights of as city one is visiting for the first time. They are also useful for preparing hypermedia presentations of existing material.

2.4 Search Facilities

Hyper-G offers two modes of searching:

- Every Hyper-G object has a set of associated attributes (title, keywords, author, creation time, expiration time, etc.), which can be searched for, including boolean combinations. Typical queries might be "Give me all documents with *fractal* and *compression* in the title", or "Give me all objects which have been created since yesterday" or "Give me all images with the keyword *clinton*".

- Text documents are automatically full text indexed on insertion, which supports the use of content-based full text queries. The Hyper-G full text server supports fuzzy boolean queries as described in [14] and WAIS-like nearest-neighbour searches based on the vector space model [15].

In both modes, the scope of the search can be defined in a flexible way:

- a single collection on a single Hyper-G server;
- a set of collections on a number of Hyper-G servers;
- the local Hyper-G server (all collections);
- a number of Hyper-G servers (all collections);
- all Hyper-G servers worldwide (all collections).

2.5 User Interface Considerations

We do not expect a user to choose one of the above navigation paradigms and then stick to it. Rather, we expect that users will want to readily change their means of navigation. For example, when looking something up in an encyclopedia, one may first use hierarchical navigation to locate a suitable encyclopedia, then issue a search on titles or content restricted to that encyclopedia, looking at the list of results in turn, and then following hyperlinks (cross references) as necessary. It is the responsibility of the Hyper-G user interface to ensure that users may change paradigm whenever they want, without ever becoming lost.

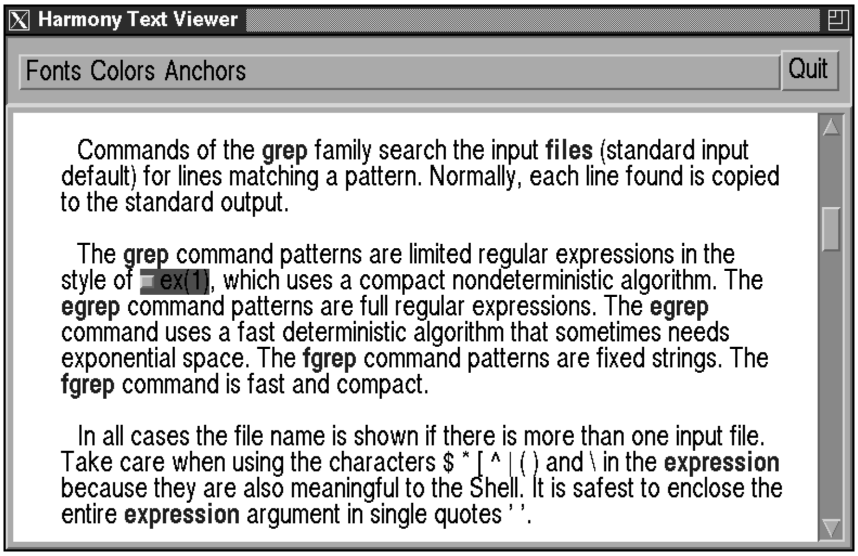

Figure 3: Harmony Text Viewer Showing Document "grep(1)"

To illustrate some of these issues, let us look at some screen shots taken from the *Harmony* Hyper-G client for X Windows. Let us assume our user wants to look up the manual page of the UNIX *grep* command. To do this, the user would activate the "Manual Pages" collection in the local server, and look for objects with "grep" in their title (the search dialog is not shown for space reasons). Figure 3 shows the Harmony *Text Viewer* displaying the resulting text document. The user could now for example follow a link to the *ex* command to look up the syntax of regular expressions. This is a standard hypertext feature found in many systems. Analogously, the Harmony *Image Viewer* allows the activation of links attached to (parts of) image documents.

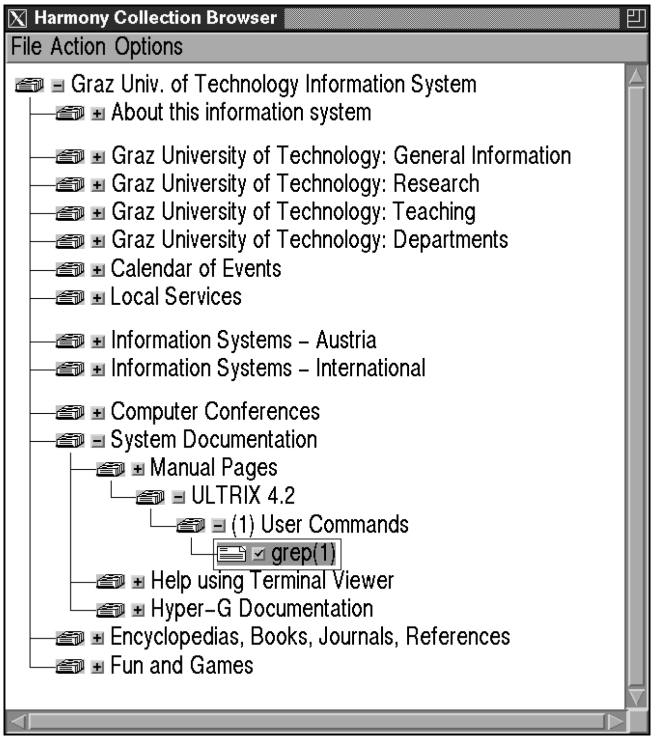

Figure 4: Harmony Collection Browser Showing Location of "grep(1)"

Rather important in overcoming the "lost in hyperspace" problem is that the location of the "grep(1)" document in the collection hierarchy is simultaneously displayed in another window, Harmony's *Collection Browser* – shown in Figure 4. The user always sees the location of the current document (highlighted by colour) with respect to the collection hierarchy, i.e. if the user now follows the link to the *ex* document, the position of that document would also be shown. In our example, we see that the "grep(1)" is part of the "(1) User Commands" subcollection of the "ULTRIX 4.2" collection. If it were also a member of other collections, all the paths from the root collection down to the document would be opened up.

The collection browser is itself active: clicking on a collection "opens" it, i.e. its "children" are shown. In our example, if the user now clicks on "(1) User Commands", the titles of all user commands under ULTRIX 4.2 are displayed. The collection browser is a kind of "global map" showing the relative position of the current document.

The collection browser shows only the collection hierarchy; it does not reflect the link

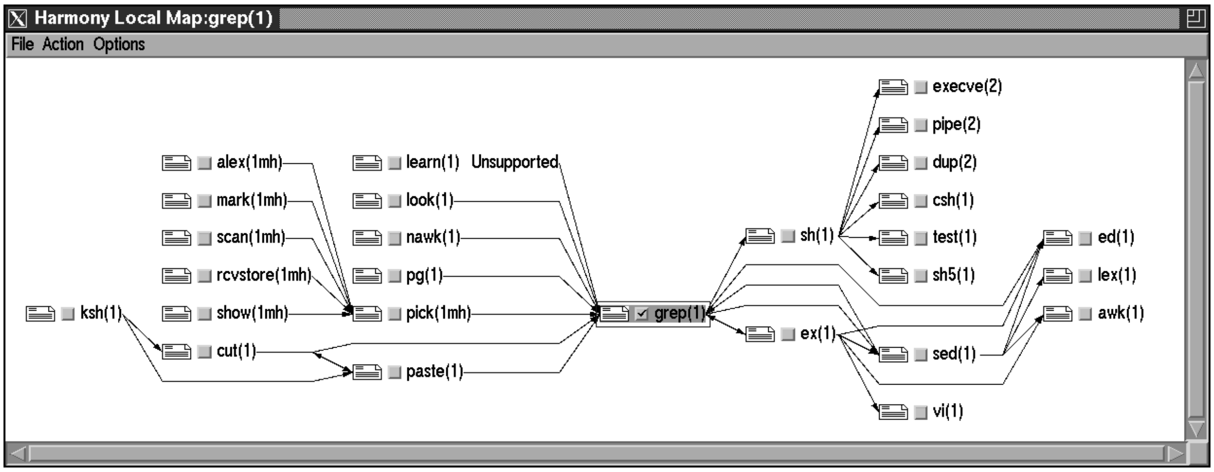

Figure 5: The Harmony Local Map Around "grep(1)"

structure. Unfortunately, in a large-scale system, it is not feasible to automatically generate a global map showing all hyperlinks. It is however possible to generate a local map (also called a web view or fisheye view) of the "vicinity" of the current document, i.e. documents related (recursively) to the current document by incoming or outgoing links. Figure 5 shows the link structure of the ULTRIX manual pages around the "grep(1)" document, two levels in either direction.

3 Architecture

This section gives an overview of the main components of Hyper-G. For more detail, the interested reader is referred to [9, 11]. The dual requirements of a large number of (multimedia) documents and a large number of users lead naturally to the use of a client/server design model, with clients and servers connected via the Internet (using TCP/IP).

Figure 6 shows the architecture of Hyper-G. Unlike Gopher or WWW clients, Hyper-G Clients are not required to connect to multiple Hyper-G servers. Rather, clients talk to the same server all the time. Should information from a remote server be needed, the local server fetches it and delivers it to the client. This approach offers the following advantages:

- It keeps clients simple and allows for an efficient, connection-oriented protocol.
- It enables caching of remote information in the local server.
- It simplifies maintenance of user accounts and access rights in the local link server (the user has to identify to one server only).
- It enables the link server to gather statistics and user profiles on a per-session basis.

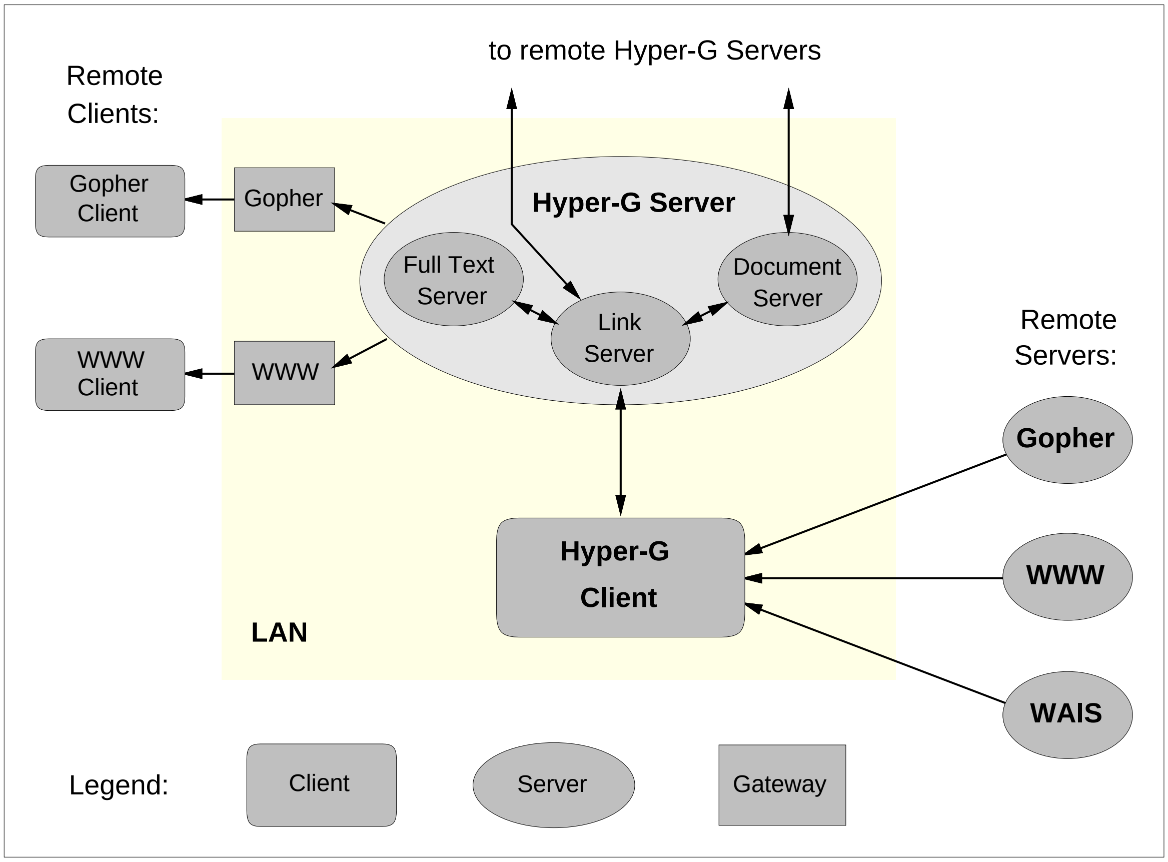

Figure 6: Hyper-G Architecture

The local Hyper-G server connects to other Hyper-G servers on demand, maintaining consistency of the information base and performing searches across server boundaries. For the client, the existence of servers other than the local server is not visible; the local server performs like a "super-server" knowing about all the information stored in all other servers. In a sense, it behaves like a domain name server which can be queried with the *gethostbyname()* system call on UNIX machines, knows about some local names, asks other servers for remote names, and caches the results for higher performance – all transparent to the calling process.

The Hyper-G server, conceptually a single process, in fact comprises three distinct server processes: full text server, link server, and document server. The *full text server* is dedicated to full text retrieval, document clustering, and automatic link generation and is not discussed further here.

The *link server* is a sophisticated object-oriented database of *objects*, i.e. descriptions of documents, links, anchors, collections, tours, remote databases, etc. (for an in-depth description of Hyper-G objects and features see [8, 10]) and *relations* between such objects (for example, which anchors are attached to which documents, which source anchors are connected to which destination anchors/documents, which documents belong to which collection, etc.). The main functions of the link server are:

- It assigns object IDs to objects and ensures that no two objects share the same ID. In addition, it guarantees that when an object is modified, it receives a new object ID so that it can be distinguished from the old version. When an object is deleted, its ID is not reused.
- It maps object IDs to objects. In Hyper-G, an object ID is just a unique number (similar to an ISBN number or a mail message id) assigned to every object (and hence every document). In the link server (and only there) more information on the object is stored: such as title, author, creation date, and in the case of documents also the data necessary to retrieve the document from a document server. Therefore, should a document be changed (say, moved from server A to server B) only the information in the link server has to be updated.
- Unlike many other hypertext systems (for example, WWW), Hyper-G strictly separates links from documents. This allows the attachment of links to documents even when the document itself cannot be changed, for example because it resides on a CD-ROM or on a remote server where the protocol prevents writing of documents. In general, this is a desirable feature since it allows users to annotate material that is otherwise read-only (for example, encyclopedias). The link information, together with meta-information about objects (title, author, creation date, etc.) is stored within the link server, while the documents themselves reside on a document server.
- A centralised link store enables the system to support *bidirectional links*[1], i.e. to answer the question "What other documents refer to the current document?". This is important for two reasons:
  - Whenever a document is deleted or modified, the system is able to tell which other documents refer to the document in question. Hence references to nonexistent or outdated documents can automatically be identified and (possibly) removed from the web, thus maintaining web integrity. This is especially important in large multi-user systems, where the person deleting a document can not be expected to know of all the links to that document.
  - It allows advanced user interfaces to draw local maps like the one shown in Figure 5.
- The link server is aware of the collection hierarchy and uses this knowledge them to maintain database consistency.
- As every Hyper-G object contains some meta-information (title, author, creation date, additional keywords, etc.), the link server can perform complex boolean and fulltext queries (for example, "Give me documents with title containing *UNIX*, created by user *fkappe* after *93/06/01*") either over the whole database or a certain subset of the collection hierarchy. Unlike Gopher or World Wide Web, Hyper-G does

[1]It should be noted that the link server concept and bidirectional links were first implemented in the Intermedia system [7].

not require the information provider to set up a search engine for every suitable collection of documents. Rather, every document and every collection is automatically searchable on creation and it is the user's decision as to where a search seems reasonable.

- A sophisticated, hierarchical *access control* scheme built into the link server supports the restriction of access to individual documents and collections to certain groups of users. The link server also supports modification of the database (including rearrangement of the collection hierarchy and editing of documents) by clients, for which access control is also a precondition. There are plans to support accounting functions as well, but these have not yet been implemented.
- The link server is also the ideal place to gather detailed *statistical data* about system usage.

Hyper-G clients connect to the link server and use it to search and browse through the information space. Whenever a document (text, image, sound, ...) is needed, it is retrieved from the *document server*, which stores all local documents and caches remote documents. All document requests are routed through the local document server. If the document requested has not yet been cached, the request is forwarded to the document server on which the document resides. The local document server retrieves the document from the remote document server and simultaneously retransmits it to the client and stores it in the cache. If the document is found in the cache, it is transmitted directly to the client.

The document server should be configured to control a certain amount of mass storage (say a few hundred megabytes of hard disk) and use it as cache memory for incoming documents. When this space is exhausted, the server removes the least recently accessed document from the cache. It is intended that the document server reside on the same LAN as the client, so that transmission from cache to client is reasonably fast.

A problem typically associated with caching in distributed environments is that of modification of what is cached. When a document is modified, it must be guaranteed that the user sees the new version of the document the next time it is retrieved and not an old copy still in the cache. Other systems do this by assigning expiry dates to objects, or by notifying all caches whenever objects are changed or deleted, which in turn requires the maintenance of a list of all caches, what they have cached, and so forth. Fortunately, however, the problem does not arise at all in our design. Remember, the link server guarantees that new object IDs are assigned to modifications of objects and that IDs of deleted objects are not reused. When a document is modified, the new version receives a new ID which is passed to the client when the user visits that document. Therefore, it is impossible that the local document server finds the new object in its cache and will automatically reload the new document from the remote document server. An old copy of the document residing in the cache cannot be accessed any more and will therefore eventually be deleted because of the least-recently-used strategy of the cache.

4 Interoperability

As was mentioned at the very beginning of this paper, we consider the interoperability of information services to be a very important issue. As indicated in Figure 6, Hyper-G is able to interact with Gopher, WWW, and WAIS servers and Gopher and WWW clients.

As shown on the left hand side of Figure 6, the Hyper-G server can be accessed using Gopher or WWW clients. When accessed by a Gopher client, the collection hierarchy is mapped into a Gopher menu tree; hyperlinks cannot be represented in the Gopher metaphor. At the foot of each Gopher menu, a synthetic search item is generated which allows the user search the corresponding collection. When accessed by a WWW client, each level of the collection hierarchy is converted to an HTML [3] document containing a menus with links to other sub-menus. The menus are marked as searchable. Hyper-G text documents are transformed into HTML documents (including their links).

In the other direction (the right hand side of Figure 6), Hyper-G clients can contact Gopher, WWW, and WAIS servers in order to retrieve information from them. The Hyper-G server is able to store pointers to such remote objects. This allows the incorporation of information on remote non-Hyper-G servers (almost) seamlessly: Gopher menus are transformed into Hyper-G collections, WWW text documents into Hyper-G text documents, and WAIS queries and responses into Hyper-G queries and responses.

In the future, we plan to move the knowledge of external protocols from the Hyper-G client to the document server. To access information stored within external databases, clients may then connect to the document server which retrieves and caches the document. In addition, the client may request the document in a specific representation (say, a certain image format and/or quality), and the document server will convert it and cache the result (and possibly also the original representation). This approach will make clients simpler, relieving them from the burden of understanding other information retrieval protocols and file formats. Software maintenance will also become easier, because only the document server's code has to be modified in order to support new protocols and file formats (remember, there may be a large number of clients tailored to different platforms and user types).

5 Acknowledgements

Partial Support of the Hyper-G project by the Austrian Ministry of Science, Joanneum Research, and the European Space Agency is gratefully acknowledged.

References

[1] Alberti B., Anklesaria F., Lindner P., McCahill M., and Torrey D.: "The Internet Gopher Protocol: A Distributed Document Search and Retrieval Protocol". March 1992. Available by anonymous ftp from boombox.micro.umn.edu in directory pub/gopher/gopher_protocol.

[2] Berners-Lee T., Cailliau R., Groff J., and Pollermann B.: "World-Wide Web: The Information Universe". *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, Spring 1992.

[3] Berners-Lee T. and Conolly D.: "Hypertext Markup Language (HTML)". June 1993. Version 1.2. Available in hypertext on the World Wide Web as http://info.cern.ch/hypertext/WWW/Markup/HTML.html.

[4] DeYoung L.: "Linking Considered Harmful". In Rizk A., Streitz N., and André J. (editors), *Hypertext: Concepts, Systems and Applications; Proc. ECHT'90*, pages 238–249, Cambridge University Press, 1990.

[5] Edwards D. M. and Hardman L.: "Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment". In McAleese R. (editor), *Hypertext: Theory into Practice*, pages 105–125, Blackwell Scientific Publications Ltd., 1989.

[6] Gloor P. A.: "CYBERMAP: Yet Another Way of Navigating in Hyperspace". In *Proc. Hypertext '91*, pages 107–121, ACM, December 1991.

[7] Haan B. J., Kahn P., Riley V. A., Coombs J. H., and Meyrowitz N. K.: "IRIS Hypermedia Services". *Communications of the ACM*, 35(1):36–51, January 1992.

[8] Kappe F.: *Aspects of a Modern Multi-Media Information System.* PhD thesis, Graz University of Technology, Austria, June 1991. Also available as IIG Report 308; IIG, Graz University of Technology (Jun 1991), and by anonymous ftp from iicm.tu-graz.ac.at in directory pub/Hyper-G/doc.

[9] Kappe F.: "Hyper-G: A Distributed Hypermedia System". In Leiner B. (editor), *Proc. INET '93, San Francisco, California*, pages DCC–1–DCC–9, Internet Society, August 1993.

[10] Kappe F., Maurer H., and Sherbakov N.: "Hyper-G – A Universal Hypermedia System". *Journal of Educational Multimedia and Hypermedia*, 2(1):39–66, 1993.

[11] Kappe F., Pani G., and Schnabel F.: "The Architecture of a Massively Distributed Hypermedia System". *Internet Research: Electronic Networking Applications and Policy*, 3(1):10–24, Spring 1993.

[12] Maurer H., Kappe F., Sherbakov N., and Srinivasan P.: "Structured Browsing of Hypermedia Databases". In Grechenig T. and Tscheligi M. (editors), *Proc. VCHCI '93, Vienna, Austria*, pages 51–62, Springer, LNCS 733, September 1993.

[13] Nelson T. H.: *Literary Machines (Edition 87.1).* The Distributors, South Bend, IN 46618, USA, 1987.

[14] Salton G., Fox E. A., and Wu H.: “Extended Boolean Information Retrieval”. *Communications of the ACM*, 26(12):1022–1036, December 1983.

[15] Salton G., Wong A., and Yang C. S.: “A Vector Space Model for Automatic Indexing”. *Communications of the ACM*, 18(11):613 ff., November 1975.

[16] Stein R. M.: “Browsing Through Terabytes – Wide-area Information Servers Open a New Frontier in Personal and Corporate Information Services”. *Byte*, 16(5):157–164, May 1991.

[17] Stubenrauch R., Kappe F., and Andrews K.: “Large Hypermedia Systems: The End of the Authoring Era”. In *Proc. ED-MEDIA 93, Orlando, Florida*, pages 495–502, AACE, Charlottesville, VA, June 1993.

[18] Utting K. and Yankelovich N.: “Context and Orientation in Hypermedia Networks”. *ACM Transactions on Information Systems*, 7(1):58–84, January 1989.