University of Southampton

Extending an Open Hypermedia System to a Distributed Environment

by

Gary John Hill

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

May, 1994

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Extending an Open Hypermedia System to a Distributed Environment

by G. J. Hill

The hypermedia approach to the structuring information has received a great deal of attention in recent years. Many authoring systems exist to aid in the creation of hypermedia documents, and many commercial applications utilise hypermedia techniques to implement facilities such as online help. However, these 'closed' systems are typically based upon proprietary formats and thus have limited wider applicability. This has led to the creation of so-called 'open' hypermedia systems which are able to offer an underlying, extensible hypermedia service to a wide range of applications.

This thesis describes the filter-based architecture of Microcosm, an open hypermedia system developed at the University of Southampton. The functionality of the system is provided in a modular manner, allowing its facilities to be easily adapted to the requirements of the users of the system, and offering a stable platform for the investigation of novel hypermedia functionality.

The extension of this model to a distributed environment is then described. The resulting system complements the flexibility of the stand-alone model with a flexible approach to the provision of distributed functionality. This allows a range of network models to be supported, from traditional client/server interaction to cooperation between users. The system is also able to support the integration of other existing distributed information services.

Contents

List of Figures

List of Tables

Acknowledgements

I would firstly like to thank my supervisor, Dr. Wendy Hall, for her support during the course of the work presented in this thesis, and for providing the opportunity to work in this area. I am also grateful to everyone else involved with the project over the years for their help and support, especially Hugh, Hutch, Ian, Mark, Nick, Rob and Steve. I am also grateful for the advice of Dr. Dave de Roure during the course of my research. Many thanks also to those people who I coerced into proof-reading earlier drafts of this thesis.

I would also like to acknowledge the financial support of Hewlett Packard during the course of my research, and Steve Gale whose enthusiasm made this possible.

Finally, I would like to thank everyone else who has helped me get through the last three and a half years: my parents for their support of my endeavours as an eternal student, and everyone who has kept me sane by dragging me away from the keyboard at frequent intervals.

1. Introduction

1.1. The Concept of Hypertext

The essential feature of hypertext and hypermedia systems, the concept of information linked in an associative way, can be seen to have existed for many years. The Talmud, a religious work, is progressively extended with annotation and commentary, whilst the early writings of Aristotle make wide use of reference to other works; similarly modern works such as encyclopaedias and dictionaries rely upon internal cross-referencing (Conklin, 1987). In the 17th century, a group of historians led by Samuel Hartlib collected a large archive (around 20,000 pages) of intellectual history. Hartlib and his collaborators shared a belief that all knowledge is interrelated ('pansophism'), and dedicated themselves to dissemination of knowledge (Leslie, 1990), their aims being very similar to those of modern hypertext visionaries. Indeed, even the source of the term hypertext has been shown to be related to the 18th century concept of 'hyperbolick space' (Rada, 1991).

The concept of using a machine to manage information structured in such an associative manner has been attributed to Vannevar Bush (Bush, 1945), a scientific advisor to Roosevelt. Bush predicted the information explosion that has occurred in recent years as electronic networks have become more widespread, and proposed his mechanical 'Memex' machine as a means of building personal information repositories. The Memex user would be able to purchase pre-prepared microfiche and build 'trails' through this and other information. However, Bush's system was never built due to the complex mechanical and information capture requirements. It was only with the advent of the digital computer that Bush's proposals were able to be put into practice. With new potential for the storage and manipulation of information, the computer offered possibilities far beyond those envisaged by Bush.

The term *hypertext* as a description of a computer system which could support associative structuring in this manner was first used by Ted Nelson (Nelson, 1967). Around the same time, the first notable attempts to develop such a system were made; NLS (oN Line System) was developed by Douglas Engelbart[1] (Engelbart and English, 1968), who was similarly influenced by Bush's work, and ZOG (which later became KMS) was developed at Carnegie Mellon University (Akscyn et al., 1988). Nelson developed his ideas for hypertext systems over many years, culminating in the concept of a global electronic publishing system which he called Xanadu (Nelson, 1987). As yet Xanadu remains an ambitious dream, although Nelson continues to strive for its realisation.

1.2. Recent History of Hypertext

Despite the long history of the hypertext concept, it is only in recent years that interest in hypertext systems has really grown. This has been attributed to the rapid increases in computer power that have taken place. Of particular note is the coming of age of graphical, direct manipulation interfaces which make it easier to interact with computer based information, and also advances in the electronic storage of information, which have helped to develop a wider use of computers in all walks of life.

With the advent of multimedia systems, the term *hypermedia* has come into use to describe the application of hypertext principles to a variety of information types. Hypertext and hypermedia systems have been proposed as an aid to a wide range of applications, from CAD systems and on-line help systems, to support for writing and argumentation processes. An in-depth discussion of the hypertext field, both concepts and actual systems, can be found in (Conklin, 1987).

The first systems to become available were dedicated hypermedia applications which required all information to be specially formatted and committed to the control of the system. Whilst such systems were able to provide a wide range of hypermedia functions, the requirement to 'own' the information they presented compromised their usability since it had to conform to the structures supported by the designers of the hypertext system. This has led to the concept of an open hypermedia system: one that is able to use data in its native form and without

[1]Engelbart is also known for the invention of the mouse as well as word processing and the use of 'windows'.

needing to embed additional information which will affect its utility outside of the hypermedia system.

1.3. The Growth of Electronic Information Delivery

A parallel development over recent years has been the increased accessibility of online information, from dial-up bulletin board systems to global networks. World-wide networks such as the Internet have grown from their roots in computer science research to encompass users of all types, using many forms of access.

A typical network user is no longer a computer researcher utilising state of the art equipment or a home enthusiast using a modem to call a bulletin board system. Many businesses now enthusiastically embrace the communication advances offered, and researchers and academics from many disciplines now use computers to access electronic information and to communicate with colleagues around the world.

Even the private computer user, who was previously restricted to accessing the 'network islands' offered by small-scale bulletin board systems, may now gain seamless access to a wealth of electronic information as the large bulletin board organisers, such as Compuserve, now provide gateways to the Internet. Newer companies exist simply to provide direct, dial-up, Internet access to companies and home users who wish to access the facilities it offers.

As the Internet user community has grown and diversified, so has the range of information available. In order to facilitate the dissemination of this information, new distributed information services have rapidly evolved to provide more flexible functionality than that offered by facilities such as FTP (File Transfer Protocol) and e-mail based servers. Using keyword searches and carefully structured information, a wide range of new services allow users to browse through widely distributed sources of information quickly, easily and often transparently.

Perhaps the clearest vision of these changes in the use of computer networks is given by the famous 'teething rings' diagram created by Nicholas Negroponte, the director of the Media Lab at the Massachusetts Institute of Technology. This diagram suggests a merger of the computer, publishing and broadcast industries as we approach the year 2000, until they are virtually indistinguishable. This diagram is reproduced in figure 1.1. The increase in awareness of computer-mediated information is just one result of this merger in action, the movement of the computer industry towards the mainstream occupied by more traditional communication media. Other changes that are representative of this change is the interest of publishing houses in the use of CD-ROM and multimedia technology, the gradual move towards digital video, and the increasing popularity of computer games.

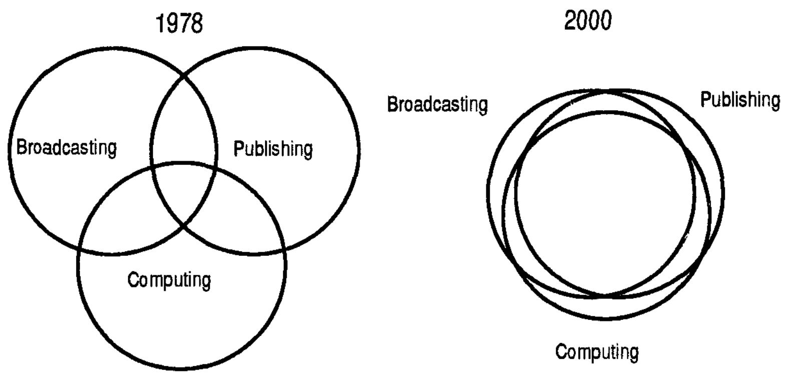


Figure 1.1: Negroponte's Teething Rings (reproduced from (Brand, 1988)).

As the growth of distributed information services continues, there will be a need for powerful tools to manage and navigate the available information. The type of facility provided by hypermedia systems is widely believed to be a method by which access to such information can be provided efficiently. As information is discovered, either by browsing a hypermedia structure, or using some other resource discovery system, it can be linked into the user's own information space and reused as required. Because the information is maintained elsewhere, there is the added benefit of access to the most up to date versions, and to cross references to information created by other users.

1.4. Overview of the Work in this Thesis

This thesis initially discusses the need for an open hypermedia system as described in the previous sections, and describes the development of the Microcosm system, an extensible, open hypermedia system.

With the expansion in the quantity of information available over the Internet described in the previous sections, there are obviously opportunities for the flexible capabilities of an open hypermedia system to be used as a method for providing integrated access to widely distributed information. After the initial description of Microcosm, the design and development of further extensions to the system are described, which are able to provide a distributed service, and access to other forms of distributed service. The structure of the thesis is outlined below.

Chapters two and three survey the current literature in the open hypertext field; chapter two describes theoretical aspects of open hypertext and establishes a set of guiding requirements with which to evaluate systems claiming to be open, whilst chapter three describes notable examples of open systems. The functionality of the systems described is then compared with the requirements developed in chapter two. Chapter four provides further background information, giving an overview of related work in the area of distributed information services, and of hypermedia systems that address the requirements of a distributed environment.

Chapter five provides a description of the motivation for, and the design and development of Microcosm, the open hypermedia system developed in the Department of Electronics and Computer Science at the University of Southampton. Chapter six provides a more detailed description of the development and refinement of Microcosm's Filter Management System. This is the part of the system which is responsible for providing a flexible and extensible range of underlying hypermedia functionality. The performance of the initial implementation of the Filter Management system is examined and certain limitations identified.

The enhancement of the Filter Management system to address these drawbacks is described in chapter seven. The performance of the resultant system is compared to the previous results obtained in chapter six. In particular, the system has been modified to support the development of a flexible distributed version of Microcosm which is better able to meet the requirements of an open system than the current generation of distributed systems which are discussed in chapter four. The distributed version of the system is described in chapter eight.

Finally, chapter nine describes some of the potential areas of future research that have arisen from the developments described in this thesis, and chapter ten provides a summary of the work that has been carried out and presents the conclusions that have been drawn.

1.5. Declaration

The work in this thesis has been carried out within a collaborative research group. It is all the author's own work, with the exception of that described in sections 5.2, 5.3.1.2, 5.3.2, 5.3.4.1 and 5.3.5, where other members of the research group have been involved.

2. The Open Approach to Hypermedia

2.1. Why we Need Open Hypermedia Systems

The current generation of hypermedia systems, developed from the early ideas of visionaries such as Vannevar Bush (Bush, 1945) and Ted Nelson (Nelson, 1967), has been extensively documented (Conklin, 1987; Fersko-Weiss, 1991). However, as the popularity of hypermedia systems increases, the requirements of the user population become more demanding and adventurous. Users are beginning to expect more than 'hyper-linked' static reference material. Unfortunately, most readily available existing systems are not well suited to handling more than a restricted body of text and graphics, bound together by a fixed set of links.

These systems generally exist as separate applications with their own proprietary document formats. If the system may be extended to allow new media types, it can generally only be done in a unidirectional manner. The hypermedia system is able to start applications, but these act as slaves, with no real way to communicate with the hypermedia 'engine' and therefore cannot perform as a fully integrated part of the system. Additionally, altering low level aspects of the system such as the basic linking model is impossible. Most systems offer only a simple, fixed linking model based on 'goto' links. It is however desirable to augment such linking with more structured facilities (Brown, 1988).

As a result of these characteristics, current hypermedia systems tend to exist in isolation from the rest of the host environment, for example the on-line help system provided by Microsoft Windows which does not even support paradigms such as copy and paste in a standard manner. Although the applications used to create information may offer a rich body of tools to manipulate and process the information, these become redundant once the information is imported into and embedded within the proprietary document format of a monolithic hypermedia system (for example Guide (OWL, 1987), or Hypercard (Apple, 1987)).

Similarly, if the system allows users to include additional information in a hypertext, the cost of the effort required is often high. Once the documents have been converted into the appropriate format, they must be manually linked into the existing body of information. The process of identifying the hypertext links required to incorporate the information in a useful manner becomes harder and more time consuming as the size of the existing body of information increases. However, if the new information is not incorporated into the existing structure, it may be hard to find even though it may be known to be in the system, and user satisfaction with the system will suffer.

Although many people recognise the potential benefits of a hypermedia approach, attempts to apply hypermedia concepts on a large scale, for example in large engineering projects (Malcolm et al., 1991), have highlighted these problems and resulted in the development of so-called 'open' systems. These are more amenable to extensions to their basic functionality and typically act as an extension to the supporting environment and its existing tools and applications, rather than simply operating alongside them as a separate application.

This chapter reviews the research that has led to the development of such systems, including the first descriptions of the need for such systems. The varying viewpoints raised are used to identify core requirements for a fully open hypermedia system, and to examine the implications of these requirements.

2.2. The First Visions

The Xanadu system proposed by Ted Nelson (Nelson, 1987) touches on many features that are now considered to be some of the most important aspects of an open hypermedia system; for example, the need for interoperability between different applications and platforms. However, although Nelson considered Xanadu to be an open system, his view is really of a closed system, albeit widely distributed and generally available. The first real suggestions of a requirement for open hypermedia systems were given by Andries Van Dam and Frank Halasz at the Hypertext '87 conference. In his keynote address at the Hypertext '87 conference (Van Dam, 1988), Van Dam discusses the gradual development of hypertext from early systems that he was involved with such as Hypertext Editing System (HES) and File Retrieval and Editing System (FRESS) through to the development of more recent systems such as Guide and Hypercard. He then goes on to outline areas which need attention in the development of new systems. Below, for example, he identifies the limitations of closed hypertext systems and the need for a hypertext 'service' as part of the operating system.

"...Ted [Nelson] talks a lot about the docuverse, a mythical entity out there that is all inclusive and contains everything. But instead, right now we are building docu-islands; none of our systems talk to each other, they are wholly incompatible. So we are all working on the same agenda, more or less, but we can't exchange stuff; there is no exchange format, there is no universality, and furthermore, our systems are closed systems. In a sense they are making the same mistake as the all-in-one environments in personal computing. Yes they all give you a word processor and a spreadsheet editor and a business graphics package, etc., but none of them are really satisfactory. And our experience with FRESS, where we had to escape to command language, showed that it is really important to be able to go outside. So it is not enough to be able to bundle the Hypercard package with every Mac you buy. It really ought to be migrated down, become part of the toolbox, so that application programmers can take their applications and take advantage of a standard linking protocol that works within and between applications."

He goes on to make other points which further illustrate the limitations of designing a closed hypertext system to provide a full solution to a users needs. There must be the ability to customise existing functionality or add new functions to suit the particular requirements of a user or application. This point is best made in his discussion of the 'view' the user has of the available information which is shown below.

"But we don't want to put things together in such a way that there is one point of view, because if we have learned one thing from interactive tools up to now, it is that multiview is the way that people work. You can not have it just one way. We need an update to "Don't mode me in." Jim Foley and I recently came up with "Don't metaphor me in." Don't give me a little card image and say that "That's all you've got, because that's what I thought you should want for your virtual shoe box." There have got to be multiple modalities and the designers have to be able to deal with that...

... don't give me only one way of looking at things"

Although this quote stresses the presentation aspects of hypertext systems, the sentiments apply equally to the links available, and the linking paradigms used.

At the same conference, Frank Halasz presented his influential paper, "Reflections On Notecards : Seven Issues for the Next Generation of Hypermedia Systems" (Halasz, 1988). In this paper he discusses the lessons learnt from the development and subsequent use of the Notecards system developed at Xerox PARC, and uses these to identify limitations in the hypermedia systems available at that point. The seven issues he presents focus on various aspects of hypermedia systems that need to be developed and may be classified into three categories:

- Modes of access - including ways to navigate through the available information, and the ways in which information may be linked.
- Dealing with changing information - support is needed for systems where the available information changes over time, to allow new information to be incorporated into the existing system and to track the changes with versioning facilities.
- Extensibility and Tailorability - how to allow the user to tailor the basic system to suit their requirements/preferences and to add new functionality for handling new types of information or for manipulating information in new ways.

Taken together, these viewpoints provide many pointers to development directions required from new hypermedia systems, including many of the features that are now being defined as essential requirements of open systems.

These issues are summed up by Norman Meyrowitz in (Meyrowitz, 1989), where he criticises the closed nature of systems such as Notecards and Guide. Describing the widespread adoption of the 'cut, copy and paste' paradigm of today's graphical interfaces, he compares hypermedia functionality -

> *"The paradigm* [cut, copy and paste] *is so widely accepted that consumers regularly sneer at and ignore software that does not provide full cut, copy and paste support....*
>
> *.... Linking functionality must be incorporated, as a fundamental advance in application integration, into the heart of the standard computing toolboxes - the Macintosh Desktop, Microsoft Windows, Presentation Manager, NextStep etc. - and application developers must be provided with tools that enable applications to "link up" in a standard manner. Only when the paradigm is positioned as an integrating*

factor for all third party applications, and not as a special attribute of a limited few will knowledge workers accept and integrate hypertext and hypermedia into their daily work process."

Meyrowitz is now putting this approach into practice in his work with the Penpoint operating system from Go corporation (Carr, 1991).

The viewpoints discussed by these authors essentially reaffirm the early visions of Bush and Nelson of an integrated and all encompassing environment for the management of information.

2.3. Defining Open Hypermedia Systems

Despite the well argued points raised by the above authors and others, until recently few systems have seriously attempted to provide the suggested functionality. This has led to a more detailed examination, by various authors, of the issues involved in the development of such systems, in an attempt to develop an agenda for the development of generally applicable, open, hypermedia environments. The following sections discuss this work.

2.3.1. Broadening Hypermedia Principles

In their paper "Broadening the Scope of Hypermedia Principles" (Maurer and Tomek, 1990b), Hermann Maurer and Ivan Tomek discuss the benefits of providing hypermedia functionality as a core part of an operating environment. Like Van Dam and Meyrowitz before them, they argue that, since hypermedia can be applied to many aspects of a typical environment, it is sensible for the required functionality to be available as a standard and centralised service. Short case studies of areas that would benefit are given; for example, software engineering where the development of software from specification through to documentation can benefit from the ability to follow concepts from specification through design to implementation. This would allow reviews to be carried out very easily, and also facilitates easy backtracking through the software life cycle, allowing developers to establish the basis for decisions that have been made and so on. Alternatively, when sending e-mail, links back to previous messages allow references made by the correspondents to be easily accessed, or a system used for computer aided instruction (CAI) can lead a user through a body of material, and at any time the user can break off, following links to find more information of interest.

Having established that hypermedia may be of benefit to a wide range of applications, the authors describe how it could be provided as an operating system service, thus offering facilities to the whole environment. Rather than attempting to force the hypermedia system to co-exist within the restraints of a normal environment based on a hierarchical file structure, the authors describe a system where links and graph structures are used to organise all the information stored, thus allowing multiple views of the same information depending on the context. Within this relaxed directory structure, the operations available can be extended so that attributes of files and directories allow functions such as powerful searching mechanisms, filtering of information and automatic generation of links. In addition to this application independent framework, programs are able to make use of the 'hyper-functionality' to augment their own capabilities, such as in the examples described earlier. The resulting system would represent a decisive move from the 'isolated information spaces' that are typically provided by the current generation of hypermedia systems, to a 'hyper-environment' where all information is structured and accessible through the use of hypermedia concepts.

2.3.2. Hypertext Interchange

An important issue in the development of open systems is how to share information between different hypermedia systems. In (Leggett and Kilough, 1991), some of the issues involved in interchange of hypermedia information between systems are discussed. The authors then outline the requirements of a hypertext interchange format, a format that may be used to transfer information between different hypertext and hypermedia systems. By defining an intermediate format in this way, any set of systems need only to convert between their own format and the interchange format in order for them to all share information.

There are however, a number of potential problems that may arise in this process. For example, the process of transferring from one system to another may cause features of the source format to be remapped to fit within the constraints of firstly the interchange format and then the target format. This change in the physical structure of the information, although it may not affect the actual content, or the effective linking between information items, might affect the way the information would appear if it were to be transferred back to the source format. Ideally, the change to a new format and back should be completely symmetrical. Features of hypertext information that might cause such problems are links with multiple destinations, and the existence of overlapping anchors.

Other problems exist in dealing with dangling links left by the transfer of partial information sets, and providing transparent and dynamic interchange of information as the user interacts with their own environment. Having discussed these issues, the authors present a case study of an experiment in hypertext exchange to illustrate some of the problems encountered in this area.

Related work in this area has resulted in several possible models for standard representation of hypermedia information (the Dexter model (Halasz and Schwartz, 1990), Trellis (Furuta and Stotts, 1990) and MHEG (Bertrand et al., 1992)). As hypermedia information systems become increasingly popular, models such as these are likely to become very important.

2.3.3. Industrial Strength Hypermedia

Perhaps the most stimulating examination of the issues involved in the development of open systems was presented at the 1991 hypertext conference in San Antonio (Malcolm et al., 1991). The authors work for the Boeing corporation, in an environment where collaborative groups of workers use a distributed and heterogeneous computing environment. In their paper they discuss the real world requirements, particularly in engineering environments, for an 'industrial strength' hypermedia system that could be integrated with their existing environment, and thus improve productivity and collaboration.

Boeing's particular requirements are for tools to maintain, and provide access to, an ever growing body of product information. This information is used in the maintenance of existing products, quality control, and in the design of new products. Engineers must be able to access this information, or add further information; they need to be able to annotate, exchange and link elements of the information; other people must then be able to take the results of the engineers work and incorporate it easily with their own results. This total information management has often been proposed by hypermedia developers (Akscyn et al., 1988; Shipman et al., 1989) but has ultimately been constrained by the use of closed systems which do not take account of existing environments and tools.

Although Malcolm et al. do not explicitly concern themselves with the requirements of open systems, they make the following statement concerning their dissatisfaction with current systems which clearly shows that their requirements go hand in hand with the use of the open hypermedia systems envisaged by other authors:

"The hypermedia world is a chaotic swirl of conflicting protocols, interfaces, representations and capabilities, and in our assessment, today's tools are not mature enough to handle the range of user and computing environments in large aerospace companies. Though there are a few multi-user hypermedia tools with limited collaborative authoring capabilities, most are closed systems that cannot be integrated with the universe of applications and platforms used in the industry. Limitations of current hypermedia products can be divided into three broad categories: interoperability, shared workspaces and interactive authoring."

These limitations are described briefly below.

Interoperability

Most hypermedia products are based on one platform, few can access external applications (e.g. databases) or access data created by other applications. For example, engineers using different computer systems for different aspects of their work would benefit from a hypermedia environment that was available across all the platforms used, providing a consistent user interface and access to distributed information.

Similarly, within even a single platform, the system must be able to incorporate any information that the user is able to create, but it is not sufficient to convert the information into a form that the system can handle; this prevents the information from being manipulated using standard tools and entails too great a load on the user who simply wants to be able to link to new information in a simple and unrestricted manner.

Shared Workspaces

Almost all real world applications can imply some level of collaboration between the users of the system. Even at the most basic level, for example e-mail, hypermedia functionality can bring benefits (Maurer and Tomek, 1990b). At a more sophisticated level, information could be brought to a desktop conference by simply selecting a keyword to branch from, or a possible design may be published into the project workspace from an engineer's private workspace.

Interactive Authoring

In an integrated information management environment, there can be no artificial distinction between reader and author. All users are equally able to add annotations and new links as they search through the available information. Thus a user can utilise all the navigation facilities that a reader may expect to have available, move quickly to a desired information item, and immediately start to create new links.

This is in sharp contrast to systems where the user must switch into a special authoring mode in order to make links (e.g. Hypercard), or systems based on mark-up languages such as Microsoft Windows on-line help, where the altered specification must be recompiled to provide new information and links.

The following requirements for an 'Industrial Strength' hypermedia system proposed by Malcolm et al. address these problems, and provide the basis of a system that could be used for general application to a real world environment.

Interoperability

As described above, the users of the system must be able to create information using any tools they wish, and incorporate it within an integrated, distributed and heterogeneous environment. This requires standards to be developed for all aspects of the system, including interchange of hypermedia data and structures, network protocols and user interfaces.

Link Flexibility

Links need to be more flexible than the simple jump provided by typical 'GoTo' links.

- Links need to have attributes for details such as author, date and restrictions on their use by other users or user groups. This allows for powerful filtering or searching based on these attributes.
- By defining different types of link to indicate relationships such as support of, or opposition to, an argument they can be used to structure information to support reasoning and decision. Alternatively, a link might provide a way to start an application, update information or control external processes.
- In order to support collaboration, there must be a concept of link ownership. A user can have private links to provide annotation to public information, or to work on a current task, whilst public links will provide access to general information.

Templates

In order to aid the migration of new information into the system, routine links should be automated. Templates allow the user to enter information with a well defined structure, such as training material and manuals, so that links can be found and created by the system. For example, a particular product plan may always be linked to specification, design, and quality control documents.

Navigation Aids

The need for navigation aids that can provide spatial awareness within a hypermedia environment is well established. In order to easily move through the available information, there must be access to bookmarks, trails, history mechanisms and other methods of browsing available information.

Configuration Control

In order to maintain the 'quality' of the hypermedia information, the ability to make and publish links must be controlled. Some parts of the system may require passwords to change the information or add new links. Such control needs to be generalised so that it can be applied to either groups or individuals; for example, in a software development environment, access privileges must be available for teams as well as individual developers. In a real world situation, versioning is important, allowing changes to be undone, and providing a development history that will be useful for quality control, reviews, and tracking decisions.

Programmability

A problem with current closed systems is that they offer limited facilities for changing the way that the information they contain is presented to the user. In future, systems must be easily extended to incorporate new functionality and to allow the presentation of information to be customised to particular user requirements. New applications must have easy access to hypermedia functionality in order to become fully integrated into the existing environment.

Storage and Networks

As the demand for new media grows (e.g. full motion digital video, high quality sound), the availability of storage and network performance become important issues. The sheer scale of storage needed to support this requirement means that very large storage devices need to be used. Since it is likely that this information will be shared, a network must be capable of delivering it at an appropriate rate since the target system may not have local storage where the incoming data stream can be buffered.

Such issues, and the fact that they are raised by potential users rather than researchers in the field, point clearly to the demand for powerful hypermedia systems and the areas that must be addressed in their future development.

2.3.4. Seven Issues Revisited

In his closing keynote address at Hypertext '91 (Halasz, 1991), Frank Halasz examined the progress made in the field since the publication of "Reflections on Notecards..." (Halasz, 1988) with respect to the issues identified in that paper. He shows that while much progress has been made on some of those issues, such as search and query (Egan, 1991), use of composites (Halasz and Schwartz, 1990; Parunak, 1991; Marshall et al., 1991), virtual structures (Gloor, 1991; Bieber, 1991), and computation (Marshall et al., 1991; Nanard and Nanard, 1991) the others (versioning, collaborative work) have not been fully investigated. The issue of tailorability and extensibility has been addressed by numerous approaches including scripting languages (Nanard and Nanard, 1991), templates (Catlin et al.,

1991) and models (Garzotto et al., 1991). However, the advent of open systems has changed the understanding of extensibility in hypermedia systems, so that the emphasis is more firmly on the provision of open protocols for interfacing new elements, rather than the provision of scripting facilities.

In seeking to establish the reasons why hypermedia has still to become a ubiquitous element of our computing environments, Halasz outlines the wide range of application areas in which hypermedia systems can be utilised, and outlines additional issues to be addressed in the development of systems which are capable of supporting these applications. The key issues raised are summarised below:

Ending the Tyranny of the Link

Halasz describes how systems such as Superbook (Egan et al., 1989) have shown that it is possible to have a hypermedia system without forcing the use of explicit links. Superbook uses 'rich indexing' of available texts to produce a powerful text retrieval package which forms the basis of information traversal in the system. This illustrates that by broadening the spectrum of linking methods used, hypermedia systems can be made more powerful, incorporating new information may be made less time consuming and browsing through information may present less of a cognitive load. New data models for linking may draw from areas such as information retrieval, expert systems or databases.

Open Systems

Halasz states that monolithic, closed systems are no longer viable, and that they are being replaced by open systems where independent processes cooperate to build up hypermedia functionality. In order to do this effectively, he suggests that researchers must examine how hypermedia functionality can be decomposed into separate processes, how these separate elements should communicate, and whether use can be made of existing object management systems such as OLE[1] from Microsoft and New Wave from Hewlett Packard or of emerging systems from members of the OMG[2] .

---

[1] Object Linking and Embedding

[2] Object Management Group

Very Large Hypertexts

Hypermedia supporters often claim that large bodies of information can be effectively managed using hypermedia tools. However the definition of a 'large' hypermedia document collection can vary from a static collection of documents supplied on a CD-ROM to the requirements of a company wide hypermedia infrastructure (Malcolm et al., 1991). In order to cope with systems of this size, the problems that must be addressed include scalability of solutions, user disorientation within a large information space, heterogeneous and distributed computing environments, conversion of existing documents and gathering of new information.

User Interfaces for Large Information Spaces

A related problem is the provision of a suitable interface to a large body of hypermedia information. Many authors have described the problems to be encountered in navigating through hypermedia documents, yet interfaces must be developed that will allow users to deal with such large graph structures. These must be adaptable to a full range of display types (portables, personal computers, workstations etc.). This may require use of novel interface concepts such as fisheye views (Furnas, 1986), or 3D models (Card et al., 1991), as well as powerful search mechanisms in order to locate an appropriate starting place in the hypermedia information.

Standards

In order to encourage the migration of hypermedia into everyday use, standards must be developed in many areas. For example anchoring of links in applications, link services, and interchange between systems. These standards must be capable of adapting to new requirements, whilst making use of existing or emerging standards in related areas that may aid in the development of open systems, for example distributed object management (Osher, 1992).

It is likely that these issues and those raised by Malcolm et al. will provide a requirements benchmark for the next generation of hypermedia environments.

2.4. Hypermedia Standardisation

As systems such as those envisaged in the previous sections become reality, standardisation will become increasingly important in order to ensure the possibility of interoperability between alternative systems. The most significant attempt to address the standardisation of hypermedia systems was the workshop sponsored by the American National Institute of Standards and Technology (NIST) at the beginning of 1990 (NIST, 1990). A number of significant works were presented at this workshop including the Dexter Reference Model (Halasz and Schwartz, 1990) and the HyTime standard (Newcomb, 1990). Other standards work is being carried out by the International Standards Organisation (ISO). These standardisation efforts are described below.

2.4.1. Dexter Reference Model.

Perhaps the most well known paper to emerge from the workshop was the description of the Dexter Reference Model presented by Halasz and Schwartz (Halasz and Schwartz, 1990). This is a formal description, in the Z specification language (Hayes, 1987), aiming to capture the important common aspects of hypermedia systems. This specification may then be used as a basis for comparison of different hypermedia systems, by describing them in terms of the Dexter model.

The model consists of three layers representing different components of a hypermedia system. These layers are shown in figure 2.1. The model focuses mainly upon the storage layer which provides a model of the node/link networks which are the basic constructs of hypermedia systems. Nodes are referred to as 'components' by the Dexter model in order to avoid any preconception that the use of the term 'node' may create. These components are treated as opaque objects by the storage layer, and thus no distinction is made between different types of components (e.g. text and graphics).

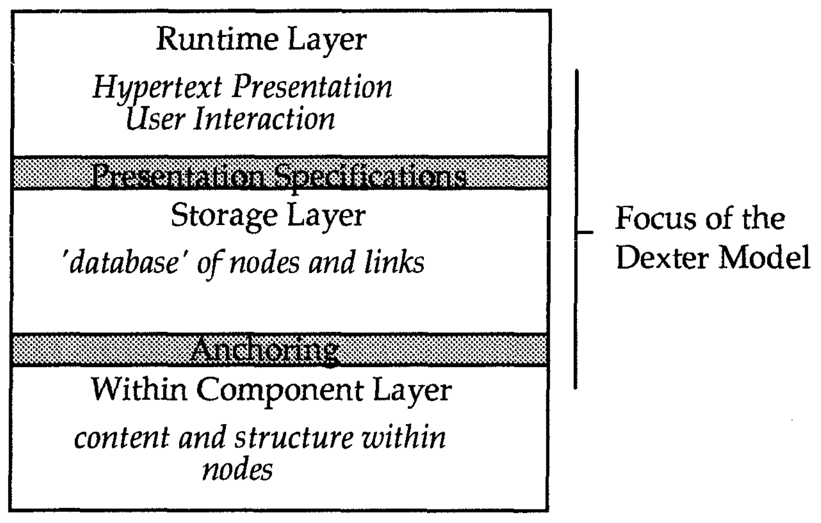

Runtime Layer
Hypertext Presentation
User Interaction
Presentation Specifications
Storage Layer
'database' of nodes and links
Anchoring
Within Component Layer
content and structure within nodes
Focus of the Dexter Model

Figure 2.1 : The Layers of the Dexter Hypertext Reference Model

Content of components not owned by the hypermedia system is addressed by the 'within component' layer of the model. This leaves the possible contents open-ended, but can be viewed as a weakness of the model (Grønbæk and Trigg, 1992), since it is unable to provide methods to deal with structured documents. The model is able to handle 'encapsulation' of these external components as a whole, for example executing a word processor to display the component, but cannot easily highlight a particular aspect of the component.

The final part of the model is the 'runtime layer' which provides tools to deal with the network structure defined by the lower levels. Again, this part of the model is left open-ended rather than limiting the possible tools. The most important aspect is its interface to the storage layer which captures the essential elements of dealing with the hypermedia components without restricting the details of user interaction.

2.4.2. HyTime

The HyTime standard (Newcomb, 1990) has developed from efforts to define a music description language. This language required the development of a method of representing links between elements of the information described, and the general approach developed has grown into the HyTime standard for hypermedia representation. The Standard Music Description Language (SMDL) is now described wholly in terms of HyTime. Both of these standards are based on the SGML[1] standard for document description.

Since it is designed as a genuine standard in which to store documents, rather than as a reference model like Dexter, the HyTime specification includes details of a wide range of link types, and document components. The nature of SGML and hence HyTime make the standard more suitable for describing static hypermedia documents rather than the dynamic and extensible hypermedia environment provided by open systems.

2.4.3. MHEG Standard

The MHEG[2] group is developing an ISO standard (Bertrand et al., 1992) called "Coded Representation of Multimedia and Hypermedia Information Objects" (ISO CD 13522, CCITT T.171). The standard will consist of two parts, a base notation, and an alternative notation using SGML.

The aim of the MHEG standard is to provide a coding of hypermedia objects so that they can easily be exchanged within and between applications. However the objects are not intended to be revised or edited, and MHEG is therefore more suited to information delivery systems rather than as a basis for a flexible, open system.

2.5. Summary

Although the exact viewpoints of particular authors vary, all of the work described in sections 2.2 and 2.3 above can be seen to share a common goal. That is, to change the direction of hypermedia system development away from the currently prevalent path where a typical system exists as an isolated application alongside other applications (for example, word processors and spreadsheets). Instead the hypermedia system will exist as an open framework, probably a component of the

---

[1] Standard Generalised Markup Language

[2] Multimedia and Hypermedia information coding Experts Group

host operating system. Applications can then make use of this hypermedia 'service' to create a powerful and integrated information environment.

Obviously as such systems become prevalent, there will be a need for standards to which systems will conform in order to provide interoperability between different hypermedia platforms. However the standards currently being developed such as MHEG and HyTime are oriented toward the description of a mainly static body of information and do not address the issues of dynamic interchange of hypertext information (Leggett and Kilough, 1991) that will become increasingly important.

Examining the issues outlined in this chapter, it is possible to identify a number of core requirements of an open hypermedia framework.

1. A system that will not impose additional mark-up on the raw information. Translating information into a proprietary format may cause desirable attributes of the original format to be discarded, and will make editing and updating the information difficult since it will no longer be understood by the tools with which it was created. Instead, an open hypermedia system should aid in the integration of the user's preferred set of existing tools.

2. A system in which data and system processes can be distributed across a network. A system operating on one host should be able to share hypermedia information with any other systems running on other hosts. Although the hosts attached to the network may represent heterogeneous hardware platforms, open communication protocols will allow them to interoperate transparently.

3. A system where there is no artificial distinction between readers and authors. Any user of the system should be able to annotate their view of the hypermedia information with additional links, and to incorporate their own information sources to create a personal view of the hypermedia information. Thus all users will require the ability to author new links in the system. Similarly, authors should be able to interact naturally with the information they are creating, rather than being expected to create a complicated hypermedia structure using separate tools. Therefore they must have access to the same browsing and navigation facilities as the end-users of the system.

4. A system which allows new functionality to be easily incorporated. No system can be expected to be suitable for all potential users, and similarly no system is likely to encapsulate all the functionality that may be required by specialised applications. Therefore the system must provide a straightforward method for new functional modules to be incorporated into the system. In particular, this modularity should be able to provide integration of existing tools, rather than requiring users to change their approach to suit the functionality that the hypermedia system provides by default.

5. The system will have support for the appropriate standards so that interchange of information between other systems will be possible. No one hypermedia system can expect to be adopted by all users, in all environments. Therefore, as hypermedia use increases, the need for powerful interchange standards will become increasingly urgent.

Although there are systems in existence that address some of these requirements, there are none that address them all successfully. The following chapter describes systems that address some of these issues, and in section 3.12 re-examines these 5 issues in the light of these current attempts. The provision of distributed hypermedia services is discussed in detail in chapter four.

3. Existing Open Hypermedia Systems

3.1. Introduction

The work discussed in chapter two offers a primarily theoretical analysis of the requirements and objectives of an open hypermedia system. This chapter considers systems that have actually been implemented and which are generally considered to address at least some of these requirements. The features of these systems are first described, and are then examined in relation to the issues identified in the summary of chapter two.

3.2. Notecards

The Notecards system (Halasz et al., 1987; Halasz, 1988) was developed at Xerox PARC with the aim of producing a system to support the structuring of related information items into an integrated and orderly corpus. As its name suggests, Notecards uses a simple card metaphor for its basic data representation. All information is stored on a computer representation of a small *notecard*. These notecards can then be stored in a *filebox* to represent classification of information; this provides a structure similar to the familiar hierarchical file system. In addition links can be made between notecards to represent the structure of the information. *Browsers* provide a graphical representation of the links available from a particular set of cards. These links are embedded in the card contents.

Notecards was built in a Lisp environment for Xerox workstations, and has an extensive programmer's interface to allow new types of card to be created. This interface can also be used to integrate existing Lisp-based applications into the Notecards environment. Details of this tailorability are given by Trigg et al. in (Trigg et al., 1987). If other applications cannot be updated in this way, then they are of limited use within Notecards, which 'takes over' the system in which it is running. Thus the ability to exploit and incorporate additional functionality is quite limited.

Another limitation of the system is that groups of cards are held in one *notefile* from which individual notecards are retrieved, thus it is difficult for tools with no knowledge of Notecards to access the actual information content of the system. An intermediate solution is available in the form of a *filecard* (Trigg and Irish, 1987); this is a type of card that simply refers to an external file. Thus, although connected to a user's notefile, the file may also be accessed by external tools, but without any hypermedia functions available during this time.

The aim of the Notecards system is to provide an integrated environment for the creation, structuring and analysis of information. As such all users are considered to be authors and no modal distinction between the two is made. There is no concept of shared notefiles, which restricts the possibilities for a distributed system, and no provision for supporting hypermedia standards is made.

3.3. Intermedia

The Intermedia system was developed at the Institute for Research in Information and Scholarship (IRIS) at Brown University (Yankelovich et al., 1988a; Haan et al., 1992). Operating on an Apple Macintosh, using a version of the UNIX operating system, it aimed to provide an integrated information environment such as that described by many of the authors discussed in chapter two. Like many other partially open systems it used a central database to store link information for the documents held within the system, and dedicated Intermedia applications were able to access the database in order to provide hypermedia services. A drawback of the system was that applications were required to place link markers in documents, thus creating problems when accessing information with external tools.

Although the driving concern behind the development of Intermedia was the provision of an environment for building and presenting teaching resources, a variety of additional components were developed to increase its general applicability as an information environment. For example, the InterMail application (Jackson and Yankelovich, 1991) that offers an integrated mail program which benefits from the linking facilities inherent in the Intermedia environment. Although Intermedia required the use of dedicated applications for presentation of hypermedia content rather than using existing tools, these applications could be constructed very quickly due to the object-oriented nature of the Intermedia development environment (Meyrowitz, 1986).

The system met the requirement of no distinction between author and reader by allowing links to be followed or made at any time in a non-modal way. A permission system placed on documents provided control over the ability to place links at various levels (owner, group, world) in a similar manner to file permission controls in UNIX.

As the native environment of the system was a dialect of UNIX, the system was able to readily support the distribution of information and processes on a network; however, the network model used was oriented towards local area networks and a single server.

Support for interchange standards was available using the Intermedia Transfer utility which maps information into the Intermedia Interchange Format (Riley, 1990). Systems understanding this format were thus able to convert and use Intermedia documents. The use of an ASCII-based representation meant that documents in this format could easily be edited. However, the complex file structure meant that standard tools would be unlikely to be able to make extensive use of information encoded in this format.

The applicability of Intermedia was restricted by the choice of a non standard operating system over the standard Apple Finder system. This further limited the number of general purpose tools available to provide further functionality to the user, and was a factor in the end of the project, as support for the system eventually became unavailable. Although Intermedia's own architecture was designed in such a way as to provide an extremely open system structure, the choice of development platfom negated all of these advantages.

3.4. Sun's Link Service

Sun's Link Service (Pearl, 1989), is well known as a groundbreaking example of an open hypertext system. Running on Sun workstations, in a distributed workstation environment, it consists of a link database service that performs all link management tasks centrally. Applications are able to link to a special library which provides a communication protocol between them and the Link Service (this system model is shown in figure 3.1). A notable feature of the Link Service is that all linking is managed separately from documents, without the use of mark-up.

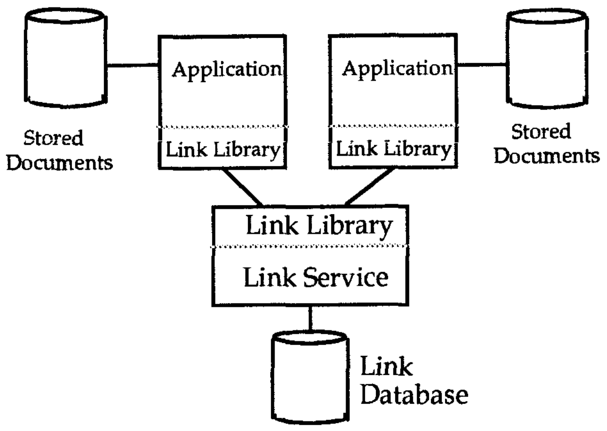


Figure 3.1 : Architecture of Sun's Link Service (reproduced from (Pearl, 1989)).

The communication protocol used is deliberately kept simple in order to preserve the autonomy of integrated tools. However, this can lead to an inconsistent user interface since each application is free to implement the required functionality in whatever way the author feels is appropriate. The linking facilities provided are also relatively simple, offering unidirectional fixed links with no additional attributes available. There are no facilities to extend the functionality of the Link Service to offer other forms of linking. This is unfortunate, since the centralised link management offers an ideal basis for an extensible system.

By using existing applications as the medium for management and viewing of linked information, the Link Service can be extended to incorporate new types of document without having to change the Link Service itself. All that is required from the existing application is the ability to identify an element in the document to link from, which can be supplied to the Link Service to form part of a link. As the body of Link Service-aware applications increases, the user would begin to enjoy an integrated environment where all of their information could be linked together.

The use of existing applications also provides a relatively transparent integration of hypermedia features into the user's environment, with authoring and browsing facilities offered in a consistent manner. However, there is no provision for the use of interchange standards to allow sharing of information with alternative systems.

Although the hypermedia facilities offered by the Link Service are relatively straightforward, the architecture of the system has set a significant precedent for the design of open hypermedia systems, and has had a great effect on the systems developed since.

3.5. Virtual Notebook System

The Virtual Notebook System, or VNS (Shipman et al., 1989), is a system designed to provide an electronic logbook for collaborative scientific work. Like Sun's Link Service, it provides a central database (built with Sybase, a commercial relational database). Unlike the Link Service however, all data objects are stored in the database rather than just the links. When the user accesses a page, the viewing application builds it from objects in the database. Links are defined upon the particular user's view so for a shared page, different users may see different sets of links depending on their requirements. This is one of the advantages of maintaining links separately from documents.

Although not a totally open system, since it requires dedicated applications to build a presentation from the hypertext database, the system has been developed further (Burger et al., 1991), resulting in a more open architecture, which does allow third parties to alter their applications in order for them to integrate with VNS. Although the underlying hypermedia functionality is also extensible within the constraints of the Sybase system, incorporating other applications at this level would prove more difficult.

The system operates in a distributed and heterogeneous environment due to the use of standards such as UNIX, X-Windows, TCP/IP, and the availability of the Sybase database system, which is used to implement the underlying hypermedia database, on a number of platforms. The distribution is based on the use of clients which communicate with a local *workgroup server*. Wider distribution is provided by allowing workgroup servers to communicate.

In common with most of the systems described, no provision for the support of standards is made.

3.6. HyperBase

HyperBase (Schütt and Streitz, 1990) aims to provide an application independent hypermedia engine. As such it is very similar to VNS with a central database providing a generic facility for storing links and data objects. Rather than build a database system from scratch, HyperBase is built (like VNS) upon Sybase, a commercial relational database system. This has the advantage of automatically providing desirable features such as heterogeneous distribution, concurrency control and security.

Because the hypermedia engine is designed to be application independent, no assumptions are made about the content of data objects held in the database; interpretation is left up to the application which is used to present the data to the user. To illustrate this, the authors also describe the SEPIA co-operative authoring system which uses HyperBase as an underlying hypermedia storage service. Applications using HyperBase in this way are able to provide their own interpretation of the contents of nodes and link specifications, but without having to concern themselves with how these items are handled by the database.

Like VNS, HyperBase fails to meet some of the criteria defined in chapter two for a system to be considered open, since it requires dedicated applications that can operate in conjunction with HyperBase to access information rather than utilising existing tools. Also, since HyperBase is designed as a general purpose hypermedia engine, it does not explicitly include the extension of the underlying functionality. It is likely however, that the SyBase system would allow such revisions to be made.

The example application presented in (Schütt and Streitz, 1990) also fails to meet the requirement that there should be no distinction between author and reader, although it is assumed that this is simply a result of the way the application is implemented rather than a feature of the HyperBase engine itself. There is also no provision for the support of appropriate standards.

The system exists as a prototype, and a full system is planned to include a Hypertext Query Language (HTQL) that will allow applications to access the database in a flexible way, thus allowing a broad spectrum of functionality to be provided by HyperBase applications. However, the flexibility of the extensions that can be developed may be limited by the strict definitions of the information supported by the HyperBase. Unlike the open ended model suggested by the Dexter model, all elements of the system are strongly defined by the HyperBase data model and could prove restrictive to some developers.

3.7. PROXHY

The PROXHY system (Kacmar and Leggett, 1991) is a prototype implementation of an extensible hypermedia architecture based on an object-oriented process model. The system is strongly orientated towards the open system approach, in that it aims to provide an underlying hypermedia service rather than a monolithic, stand-alone application.

The model supports a simple hypertext model where anchor objects within application objects are defined, and connected by link objects to provide a traditional directed graph hypertext system. Details of these links and anchors are maintained in a persistent object store, separately from the information being linked. The hypertext services to provide this functionality are provided by four process layers. This is illustrated by the diagram reproduced in figure 3.2.

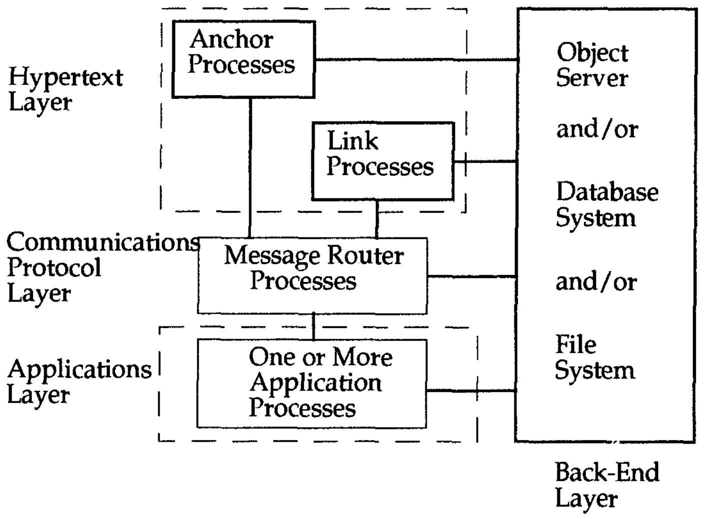


Figure 3.2 : The architecture of the PROXHY model (reproduced from (Kacmar and Leggett, 1991))

Each of the layers shown in the diagram may contain any number of separate processes. An object-oriented approach allows inheritance and message passing between processes, supported by the communications protocol layer. This generic layer is based upon message routers which provide an abstract interface to interprocess communication. A consistent interface is provided to both the objects and classes of the hypertext model, and the applications making use of it.

The message routers operate based on object definition tables which allow messages to be routed to the appropriate elements of the system. These also provide the inheritance mechanism with the aid of subclass and superclass entries in these tables. If the target object identified does not support the message type being routed, the entries in the table are used to traverse the class hierarchy, to search for a class that is able to process the message.

The hypertext layer consists of anchor and link classes that operate as separate communicating processes. These are able to operate in parallel to support hypertext activity. By extending these classes, the hypertext functionality provided can be altered and extended. For example, the system could be extended to allow the inclusion of keywords to support filtering of the available links and anchors.

The back-end layer provides storage services to the other parts of the architecture. This may simply consist of persistent object storage (of links, anchors and other parts of the system), database systems, and access to the underlying file system.

Finally, the application layer consists of processes that make use of the services provided by the rest of the system. These applications may consist of several communicating processes, as with the components of the hypertext layer, or a single monolithic application. To integrate with the system, an application simply has to understand the communication protocol and the messages which it may receive and send. These messages deal with the creation of anchors, and the type of display an application should provide. The model offers flexible support for the presentation of anchors, either allowing the application to indicate the position at which anchors should be shown, and displaying them, or allowing the application to display them itself. The message protocol is open, and flexible, with applications able to choose exactly which messages they wish to support. This makes it easier to integrate existing applications with limited scope for customisation.

The PROXHY prototype implementation of this model is implemented in C for a variety of platforms including IBM RT and SUN. The notable feature of the prototype is its support for integration of applications that have not been developed with PROXHY support. These may be incorporated to a limited extent by the use of *proxy* anchor objects. A proxy anchor records the details required to execute an external application to display a particular document, then when a link is traversed ending at such an anchor, the appropriate program can be executed. A limitation of this approach is the coarse granularity at which nodes may be dealt with; for example, highlighting a specific endpoint of a link in the document would be difficult.

The underlying object system utilises a globally unique naming system, and the message passing between processes allows the individual processes to reside on networked systems, so building a distributed system is possible. However, practical experiments have shown that the required message traffic restricts performance in a distributed environment.

The design of the system as an underlying, integrating service means that, as with Sun's Link Service and other similar systems, browsing and linking of information is essentially modeless, and there is no author/user distinction. However, no support for standards is provided. With the system composed of numerous independent processes, creating an encapsulated version of the linked information for interchange with other systems might prove difficult.

3.8. Hyper-G

Hyper-G (Kappe et al., 1992; Maurer and Tomek, 1990a) is a wide-ranging hypermedia project under development at the Technical University of Graz in Austria. It is an ambitious distributed system involving access to central resources from a range of heterogeneous workstations. The system will include a wide range of general information (such as encyclopaedias and dictionaries) as a core database, and is intended to be used as an educational tool.

In fact, the main motivation of the project is to provide a general purpose university information system, and as such it is designed to support a wide range of users. Researchers will be able to access results and publications in their field, support will be included for teaching with on-line lessons and classrooms, administrators will have access to the minutes of meetings, regulations, timetables etc., and a communication framework will encourage collaboration between users of the system.

The information stored is arranged in a hierarchy which at its lowest levels has information *chunks,* which are individually addressable items of information. Chunks are assigned keywords which may be used to generate links automatically, or to allow searches across the body of stored information.

A major obstacle to the adoption of such a grand scheme is the problem of incorporating vast quantities of existing information in the hypermedia structure. To support this the developers expect to be able to offer a system where documents can be faxed to a special server which will provide an OCR-derived electronic version of the documents. Documents received in this way would still need to be joined into the hypermedia structure, either through the use of keywords to automatically generate links, or by the user creating links.

An innovative feature of the system is the proposed support for multiple languages. Links can be generated on various keywords, but in a large system, particularly in Europe, it can be expected that documents may be available in a variety of different languages. The Hyper-G design incorporates a variety of translators to make such links automatically available whatever language a particular keyword is selected in.

Hyper-G is currently under development and obviously some of the features described are still not fully investigated. Its design meets many of the requirements for an open system, such as the separation of links from documents, and the distribution of information and processes. However, although the research team developing the system intend to experiment with different approaches to elements of the system, for example user interface metaphors, the underlying architecture is not inherently extensible and therefore may not be easily tailored to different usage requirements. The system can, however, incorporate other information services such as Gopher and World-Wide Web as remote databases.

The interface to the system is designed to be separate from the underlying hypermedia engine, therefore a variety of interface metaphors may be used. Thus an interface with no distinction between user and author is possible. There is no description of any support for interchange standards in Hyper-G. Further discussion of the distributed aspects of Hyper-G can be found in chapter six.

3.9. DeVise Hypermedia

The DeVise Hypermedia system (DHM) (Grønbæk and Trigg, 1992) is being developed at Aarhus university in Denmark, as part of the DeVise project. The aim of this project is the development of general tools to support system development and co-operative design in various application areas, in particular large engineering projects.

If a hypermedia system is to be used to integrate such tools, it must be available on all hardware platforms being used, and must be capable of being easily extended and tailored to particular requirements. Additionally, since it has to operate alongside many other tools as part of the DeVise project, DHM cannot expect to 'own' all the data which it must incorporate, but must be able to integrate third party tools where necessary. The Dexter reference model (Halasz and Schwartz, 1990) was selected as a sound basis on which to base the design and development of such a system.

Although the Dexter model proved to be suitable as a basis for DHM, the authors have identified areas of the model which require clarification and extension. In particular, the Dexter model explicitly forbids the existence of dangling links although the semantics of the model are able to describe them. Grønbæk and Trigg identify several situations where a dangling link may be justified. For example, a node may exist on a different file system that is not currently available, or may be locked for editing by another user. Alternatively a dangling link may be used to show that further information is to be added, but has not yet been incorporated into the hypermedia structure.

Another area in which the Dexter model is shown to be lacking is in the treatment of component contents. The model allows information to be owned by the hypermedia system (in 'runtime' and 'storage' layers), or by the displaying application (the 'within-component' layer). This opens the way for the integration of external applications into the hypermedia environment, but does not fully examine the additional requirements of such integration. For instance the model cannot distinguish between objects managed by the hypermedia system, and those whose contents are managed by the application as is the case with external applications.

This leads to problems in the way that external documents with their own internal structure are addressed. Although it is very easy to address the document as a whole, it is very difficult to expose the internal structure of the document for link anchoring.

The development of DHM (described in (Gronbaek et al., 1993)) has been based upon the use of a general purpose, object oriented database as a central server. This provides the physical implementation of the underlying storage layer defined by the Dexter model. A logical storage model defining the DHM-specific structure of the stored information provides an interface to this database. By extending the facilities of the object-oriented database engine, further hypermedia functionality could be developed. It is not clear whether such extensions could include the use of external applications.

For each user of the system, a runtime process co-ordinates the applications being used to display stored information, and provides communication concerning interactions and presentation between these editor processes and the database. The system is able to incorporate third-party applications, such as word processors and spreadsheets, the degree to which this may be done depending on the actual application. This leads to a consistent interface to both link creation and browsing the hypermedia information.

The distribution of the system is currently based around the use of a single server, but in future communication between multiple servers may be possible. There is no explicit provision of support for standards, but the use of the Dexter reference model as a basis for the system design suggests that the translation of the linked information would be relatively straightforward.

3.10. Multicard

Multicard (Rizk and Sauter, 1992) is an open hypermedia system developed at INRIA in France as part of the ESPRIT Multiworks project. The various components of Multicard include a developer's toolkit, authoring tools, a scripting language, a communications protocol and editors.

The aim of the Multicard system is to attempt to combine the object management properties of large scale systems such as HyperBase (Schütt and Streitz, 1990) and

HAM (Campbell and Goodman, 1988) with an open communication protocol like that of Sun's Link Service (Pearl, 1989). The provision of an easily accessible hypermedia service to external applications is intended to allow straightforward integration of tools into a powerful common hypermedia environment rather than having many applications all providing simplistic hypermedia extensions of their own. The Multicard approach is based loosely on the structure defined by the Dexter reference model (Halasz and Schwartz, 1990).

The Multicard system is based on a modular architecture where a back end provides a storage platform which is utilised by the front end applications. Communication between these elements takes place through the use of the M2000 communication protocol. This protocol defines a selection of messages that may be sent to applications, or from applications to Multicard. For instance, Multicard is able to instruct applications to open and close nodes, adjust available menu options, and select areas. Applications may support any set of Multicard messages that the developer deems appropriate to the functionality of the application. Similarly, applications may send messages to Multicard, these may include replies to the messages described above or notification of events such as mouse and menu actions.

Although the Multicard protocol is more extensible and flexible than that offered by other systems, such as Sun's Link Service, the requirement for applications to be extended is still a limitation which restricts the applicability of the system. Additionally, although the hypermedia structure is navigated using applications which comply with the M2000 protocol, this structure must be created using a separate authoring tool. This clashes with the requirement for consistency between author and user interfaces.

The scripting facilities of the system offer some ability to tailor the functionality of the system, but this happens on a per-object basis, rather than as a generic extension of the base functionality of the system.

The underlying object store is able to operate in a distributed environment, but it is not clear whether this could be extended to include inter-server operations, or if it is restricted to the provision of a single central server. There is also no provision for the use of standards.

3.11. Hyperform

The Hyperform system (Wiil and Leggett, 1992), is a hypermedia system developed in a collaboration between Aalborg University in Denmark, and Texas A&M University. The system is intended to address the weaknesses of traditional 'hyperbase' systems, or 'the HAM generation'. These systems offer strong but fixed support for what their developers consider to be the appropriate general functionality of a hypermedia system. However, this functionality and its underlying data model cannot easily evolve as the requirements of users develop over time.

Learning from their own experiences in this area, the authors have developed Hyperform, a system which aims to provide a tailorable and extensible hypermedia system. The Hyperform system is implemented using Elk (Extension Language Toolkit) which is itself based on the Scheme language. Elk also allows the incorporation of features written using other languages by dynamically loading object files. Using this system, Hyperform is implemented as an extension to the Scheme language. It implements basic hyperbase services that may be used to build specialised hypermedia support. This is done by providing a small class hierarchy of object types, which may be specialised using inheritance. These classes offer methods to support concurrency control, notification of events, access control, version control, and search and query functions. This allows the underlying hypermedia functionality to be extended quickly and easily.

The interface to the services provided by Hyperform is provided by 'Tool Integrators'. These are able to incorporate internal tools developed using Elk, or external tools via the UNIX socket and shell interfaces. Obviously, the extent to which external tools can be incorporated depends a great deal on the flexibility which they offer to extend their own functionality.

The authors have demonstrated the flexibility of the Hyperform approach by simulating the hypermedia models of both HAM (Campbell and Goodman, 1988) and the Danish Hyperbase (Wiil and Østerbye, 1990). The slight performance penalty due to the flexible approach is offset by the speed at which such simulations can be developed.

The system offers a centralised client-server system, and is therefore well suited to distribution in a local area network. However, it is not clear how well this will translate to a more widely distributed environment, especially the cooperative aspects of the system.

3.12. Summary

It is apparent from the systems described that many developers are attempting to move away from the closed system model to investigate the possibilities of an open approach. However, none of the systems described has yet managed to meet all of the core requirements for a fully open hypermedia system described in section 2.5 These issues are reviewed here with respect to the systems described above.

1. *A system that will not impose additional mark-up on the raw information. Translating information into a proprietary format may cause desirable attributes of the original format to be discarded, and will make editing and updating the information difficult since it will no longer be understood by the tools with which it was created. Instead, an open hypermedia system should aid in the integration of the user's preferred set of existing tools.*

   Most of the systems described feature this important aspect of an open system, some form of central hypertext service available for all applications to utilise in order to provide hypermedia functionality, this removes the need for links to be hard-coded into documents. Some systems however still impose mark-up on the original data to represent anchors, which are used to access the links in the database (for example Intermedia), or require full ownership of the linked information (e.g. systems based on relational databases such as VNS and HyperBase).

   Separation of link information is important as this allows linking to any information, for example read-only data from a CD-ROM, and permits alternative sets of links to be overlaid on documents depending on the requirements of particular users or the type of relationships being examined. However, the use of embedded anchors and the enforcement of a central object store counteract the benefits of a central link database, by maintaining the notion that the hypermedia system is the owner of its world and that only one link structure is required.

Many of the systems described make facilities available to incorporate existing applications, although the degree to which applications must be tailored can vary. For example Sun's Link Service provides a simple protocol that must be adhered to, and VNS provides a similar mechanism, although applications are expected to use the central hypermedia service for object storage. Older applications such as Intermedia and Notecards are less able to do this. Intermedia in particular required applications to be developed using its proprietary object hierarchy whilst Notecards was able to incorporate applications developed in its host Lisp environment. Newer developments such as PROXHY and DHM allow the encapsulation of uncooperative applications without any tailoring, as a limited but effective method of integration.

Whilst in the long term it is probable that hypermedia functionality will migrate to the level of a basic operating system service, in the short term it is important to be able to allow the use of whichever tool the user prefers. This will inevitably include applications that are not integrated with the particular system being used, and which do not provide the extensible functionality that would allow them to be suitably enhanced.

Also, it is possible that users will be in a position to choose from a variety of hypermedia engines for their particular platform. In this case, it is even more important that the system is able to integrate independent tools that have not been developed with a particular hypermedia platform in mind.

2. *A system in which data and system processes can be distributed across a network. A system operating on one host should be able to share hypermedia information with any other systems running on other hosts. Although the hosts attached to the network may represent heterogeneous hardware platforms, open communication protocols will allow them to interoperate transparently.*

Many of the systems described are based in an environment which is already distributed (e.g. Sun's Link Service, VNS and several others), and thus are able to make use of existing network facilities for accessing remote information and processes, although some offer weak support for distribution (for example Notecards does not allow concurrent access to information by groups of users). The model used however is typically based upon a strict client/server demarcation, which could prove limiting as networks increasingly consist of multiple workstations that are very powerful tools in their own right. Additionally, most do not address the possibility of multiple servers, relying instead on a single centralised service. This is restrictive when attempting to offer a more widely distributed system. These issues are discussed further in chapters four and eight.

Only VNS addresses heterogeneity explicitly. However many systems use widely supported network standards such as TCP/IP which would allow heterogeneous interoperation assuming that the system can be successfully ported to different platforms. If hypermedia is to successfully operate in a widely distributed environment (for example the aerospace industry (Malcolm et al., 1991)) then interoperability between heterogeneous platforms is crucial. The degree to which existing systems have provided distributed services is described in the following chapter.

3. *A system where there is no artificial distinction between readers and authors. Any user of the system should be able to annotate their view of the hypermedia information with additional links, and to incorporate their own information sources to create a personal view of the hypermedia information. Thus all users will require the ability to author new links in the system. Similarly, authors should be able to interact naturally with the information they are creating, rather than being expected to create a complicated hypermedia structure using separate tools. Therefore they must have access to the same browsing and navigation facilities as the end-users of the system.*

In the type of integrated environment envisaged for the future of open hypermedia systems, this is an extremely important requirement. The aim is to use hypermedia functionality to allow users to browse and access available information easily, whilst at any point it must be possible to incorporate new information, as appropriate references between documents are identified.

Nearly all of the systems described aim to provide a general hypertext service for the provision of an integrated environment. As such they are biased towards providing hypermedia support to existing applications, and as a result the modality of user/author found in many monolithic systems (e.g. the scripting mode of Hypercard or Toolbook) is removed almost by default. Users continue to utilise the same applications as they always have but now may also cross reference information available in many other applications easily.

However, in order to provide such an environment, the underlying system must be able to provide users with their own private 'link space' where any annotations or links that they add to shared information may be stored. This was a problem with Intermedia which restricted users to accessing only one 'web' at a time. Thus when browsing a shared web, a user was unable to make private additions to the hypermedia structure. Systems based on central database systems (Sun Link Service, VNS, HyperBase, Hyper-G) must be able to store ownership attributes to allow this type of functionality.

4. *A system which allows new functionality to be easily incorporated. No system can be expected to be suitable for all potential users, and similarly no system is likely to encapsulate all the functionality that may be required by specialised applications. Therefore the system must provide a straightforward method for new functional modules to be incorporated into the system.*

Although most of the systems described offer some form of integration to existing applications, this is typically restricted to extending the range of media supported by the front-end of the system. Few systems have a similar interface that will allow new components to enhance the link services provided. This is unfortunate since there is much evidence to suggest that alternative linking paradigms can offer a valuable alternative to the more usual, authored point to point link. For example, the use of information retrieval (Li et al., 1992), set-based classification (Parunak, 1991), string search (Faloutsos et al., 1990) and expert systems (Colson and Hall, 1991).

Notable exceptions amongst the systems described are the Hyperform and PROXHY systems which provide extensible object based models, and Multicard which offers a scripting language. The drawback with these systems is that extension is restricted to the use of the supplied facilities and they do not allow the use of open protocols to incorporate third party applications in the same way as for front-end applications. This restricts the possibilities for the extension of the underlying hypermedia functionality of the system.

5. *The system will have support for the appropriate standards so that interchange of information between other systems will be possible. No one hypermedia system can expect to be adopted by all users, in all environments. Therefore, as hypermedia use increases, the need for powerful interchange standards will become increasingly urgent.*

   Of the systems described, only Intermedia considers the problem of information interchange between systems. Their solution is specialised to their own system, but since it encapsulates the data held in the system, it can be used to map the information into a standard interchange format (for example, the work of Leggett and KIllough who map the Intermedia Interchange Format into the Dexter Interchange Format specified by the Dexter reference model (Leggett and Kilough, 1991)).

   Although powerful hypermedia systems are still a relatively immature technology, there is a need for standards to encourage their adoption. Although many have been proposed (HAM (Campbell and Goodman, 1988); Dexter (Halasz and Schwartz, 1990); Trellis (Furuta and Stotts, 1990); MHEG (Bertrand et al., 1992)), there is as yet no widely adopted standard. A problem with some of these existing standards is the type of hypermedia systems that they are aimed at. For example, the MHEG and HyTime standards that are currently being adopted by international standards organisations are presentation-based and do not address all the areas that are pertinent to open hypermedia systems.

The later chapters of this thesis discuss the development of Microcosm, an open hypermedia system that has been developed in the Department of Electronics and Computer Science at Southampton University. It aims to address some of the limitations of current systems described above, in particular the integration of existing tools, and the provision of extensible core functionality. The next chapter outlines the development of distributed information services, including distributed hypermedia systems, and discusses the success with which such systems meet the requirements of an open hypermedia system from chapter two.

4. Distributed Information Systems

4.1. Introduction

The preceding chapters have discussed the requirements of an open hypermedia environment, and examined examples of hypermedia systems which meet some of these requirements. This chapter discusses the characteristics of distributed information systems, of both a general nature and those which provide hypermedia functionality. The degree to which these systems meet the requirements identified in chapter two is also described.

4.2. The Dream of a Global Information Network

From their origins in the visions of Bush and Nelson, hypertext and hypermedia systems have been proposed as a method for the structuring and storage of vast bodies of information. Bush's Memex system (Bush, 1945), described in chapter two, used microfilm as its main storage media and was designed as a powerful mechanical tool which would allow trails to be composed amongst the many rolls of microfilm to which it had access. The system he proposed would have been able to store enough information that the average user would take hundreds of years to fill the repository, even at an entry rate of 5000 pages a day. This system, however, was oriented towards the creation of personal libraries of information, and to add further information it had to be purchased on microfilm or entered by a manual, photographic process.

It took the extraordinary vision of Ted Nelson to make the connection between this proposed information system, and the possibility of using electronic computers to actually build such a system. Nelson's grand vision is of a hypermedia system that would be able to provide electronic access to any form of information. In order to bring this dream into being, Nelson designed Xanadu (Nelson, 1987), an ambitious distributed hypermedia system which is based on the concept of a world-wide array of information servers or 'feeders'. Potential users would be able to connect to the system from their homes, or by using systems in information 'stations' which provide Xanadu services to a particular location. Nelson's description of such stations brings to mind an amusement arcade, bar or similar social gathering!

Unfortunately, the extremely ambitious nature of Xanadu has meant that a functioning system has not been developed, despite the support of large computer companies such as Autodesk. There have been a number of notable hypermedia systems which have addressed the issue of providing information in a distributed environment. However, these systems have been more limited than the Xanadu design, and have not seen the widespread use foreseen by Nelson and Bush.

One factor that has created new possibilities for the creation of such a system is the rapid expansion of the Internet (see figure 4.1), from a facility used mainly by the computing departments of academic institutions, and large computer organisations, into a widely available computer network with access possible even from the home. This has led to a greater interest in the provision of distributed information services, and the development of various systems that attempt to address this requirement.

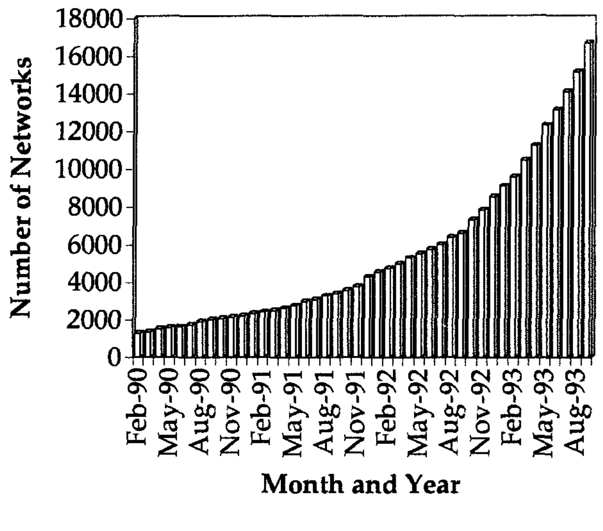


Figure 4.1: The growth of the Internet by network between February 1990 and August 1993. The graph is compiled from statistics maintained by NSF, and available by anonymous FTP from nic.merit.edu as /nsfnet/statistics/history.netcount

The following sections describe notable examples of these new information services, and examine the brief history of distributed hypermedia systems.

4.3. A New Generation of Distributed Information Servers

Recent years have seen much development in the field of distributed information servers, particularly in the Internet environment. These systems, normally based on the client/server model, allow information providers to make data accessible via servers in their own network domains. Users around the world connected to the same network can use client programs to access the information provided by these servers.

The use of client programs on powerful local workstations allows versatile interfaces to be developed and offers a great deal of device independence to both client and server platforms, which need only understand the appropriate communication protocols and data formats in order to create a heterogeneous, distributed information system. For example, client and server programs are available for most of the common computing platforms (e.g. DOS, Macintosh System 7.0, UNIX and VMS). The general model of these systems is illustrated in figure 4.2. The following sections describe some examples of such systems.

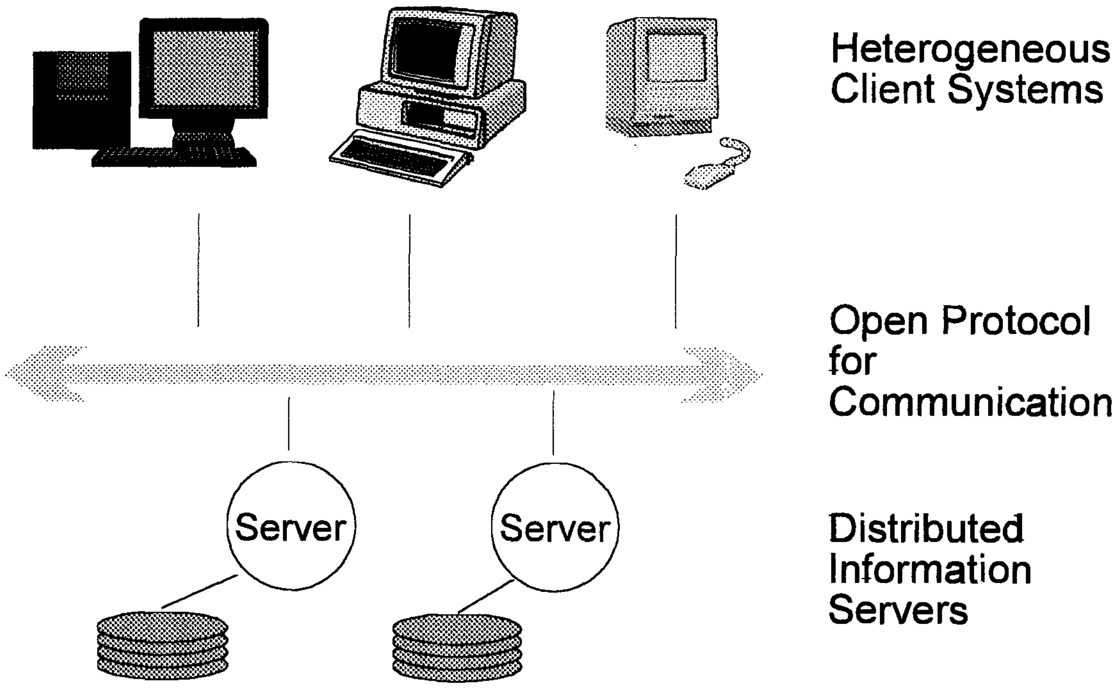

Heterogeneous Client Systems
Open Protocol for Communication
Server
Server
Distributed Information Servers

Figure 4.2: General architecture of distributed information services.

4.3.1. Wide Area Information Server (WAIS)

One of the earliest examples of such a system was WAIS (Kahle, 1989). This architecture was developed by Thinking Machines Corporation, primarily to exploit the power of their highly parallel computers as information servers (servers have also been implemented for more common architectures).

A WAIS server provides access to a list of information *sources*, each providing access to a particular type of information. The WAIS user can select appropriate sources, and specify keywords, which together form the parameters of a 'question' to the WAIS server(s). The servers supporting the chosen sources process the query by examining a full text index of the available documents, and return a list of 'hits'. These describe documents that match the keywords specified in the question. A ranking between 1 and 1000 is given for each listed document in order to guide the user to the documents which are expected to be most relevant. Additionally, some of the documents may be marked as relevant by the user, and used in subsequent searches to guide the server. This is known as relevance feedback.

In order to help users find servers that are able to provide appropriate information, a server is maintained that contains descriptions of all known servers. This can be queried in the normal way to find details of available WAIS sources. These can then be easily used to perform queries. The majority of the information available through WAIS is of a textual nature, since this can be indexed more easily, however some graphical information has been made available through the use of simple keywording.

The designers of WAIS have a vision of a massive corpus of information located on various servers, which combine to provide an information service to a widely distributed user community. Although the WAIS model does not provide any form of hypertext functionality, the similarities with Xanadu (Nelson, 1987) are clear. In particular, just as the design of Xanadu makes provisions for the payment of royalties relating to the use of material, so the WAIS model includes methods for collecting payment, thus allowing servers to finance their own support and development. Although most servers are currently provided by the academic community and currently do not use this facility, Thinking Machines are aiming to produce a commercial version of WAIS (until now the software has been freely available in the public domain), and will presumably actively encourage commercial information providers to make use of WAIS as a flexible delivery medium.

Communication between WAIS clients and servers is carried out using a tailored version of the Z39-50 ANSI protocol (Lynch, 1991). It does not at present support the boolean searches defined by the protocol, and extensions have been made to support the relevance feedback functionality described above. WAIS currently operates using TCP/IP although the Z39-50 protocol is compatible with the ISO SR protocol and hence the OSI network model.

4.3.2. Gopher

The Gopher system (Alberti et al., 1992a), developed at the University of Minnesota, acts as a fairly straightforward, distributed, file system. Each Gopher server organises its information as a hierarchy, where intermediate nodes are equivalent to directory listings in a normal file system, and leaf nodes are documents which may be viewed by the user. The Gopher hierarchy encodes the type of the various nodes so the client is able to provide an appropriate view of the document, the more sophisticated Gopher clients may be configured to use external applications of the user's choice for the various document types.

At its simplest, Gopher may be viewed as a slightly more flexible interface to document retrieval than the Internet File Transfer Protocol (FTP). However the hierarchy accessed through a particular server is able to transparently incorporate other Gopher servers allowing sites to logically arrange the information already available within computing facilities, and to link to applicable information from other server hierarchies. More sophistication is possible through the use of *search servers*. These may be used to provide a virtual node that is the result of a search over some part of *gopherspace* (the set of all Gopher servers in the world). This search facility has been used to provide a directory of servers similar to that provided by WAIS.

The gopher protocol is very simple, based on client/server interaction over a data stream (in the Gopher world at present the protocol is firmly based on TCP). A connection is established to a server, and the client sends a *selector* string which specifies the requested node. The server returns the information then closes the connection. If the requested node is a document then the information returned is simply text or binary data. For intermediate nodes, the returned information describes each line of the Gopher menu. Each line consists of the node type, the description string to be displayed, the selector string for the node, the server address, and the port number that can be used for connection.

Unlike WAIS, there is a wide selection of information available in Gopher other than text mainly due to its interactive, browsing interface as opposed to the query system employed by WAIS. These other documents include many forms of image, sounds, digital movies and software packages. A new version of the Gopher protocol Gopher+ (Alberti et al., 1992b) is being developed which will provide additional support for multimedia data. Setting up a Gopher server is very easy, and simply involves bringing together the information to be offered, and arranging it in the appropriate hierarchy.

4.3.3. Prospero

The Prospero system (Neumann, 1992) is a tool that allows users to develop their own organisation of distributed information, in contrast to the static views that may be browsed by users of the Gopher system. The Prospero model consists of a directed graph connecting distributed directories. A user may build their own virtual file space by building *views*. These are directories derived from various sources, such as the user's own information, and views from various Prospero servers. Servers may simply supply directory listings, or the results of searches across some part of the Prospero graph space.

The user is provided with a set of tools for traversing their virtual space in the same way as a traditional file system (where one would use *cd* and *ls* in a UNIX environment, Prospero offers *vcd* and *vls*). Rather than simply traversing a file hierarchy, these commands must perform a distributed computation over the Prospero servers represented by a particular view.

Prospero operates over a variety of existing network file systems, including Sun's Network File System, the Andrew File System and anonymous FTP. The links made by a user to form their views include the server from which the necessary information may be retrieved and the appropriate access mode. To publish information in the Prospero system, a user may register their system with the server administrator who will create a link to the new system in the Prospero master view, where other users will be able to access it.

4.3.4. Other Systems

There are also a number of other distributed information systems in operation, offering more specialised services to users. These are listed briefly below:

Archie (Emtage and Deutsch, 1992) - This is a database service that allows the user to search for programs in FTP archives on the Internet. The user simply supplies a string to search for, and the Archie system returns a list of matching files and their location in an FTP site. The database system is not truly distributed, it simply co-ordinates itself through a master site which produces the master database from registered FTP sites.

Netfind (Schwartz, 1990) - This is an e-mail address directory equivalent to the white pages telephone directory. Given a user name and organisation, it attempts to locate information about the user. Netfind operates by monitoring user and host information in sources such as Network News and the Domain Name System (DNS), and uses it to build a database. Given a particular organisation, the Netfind system selects the most appropriate host names from the database. Using DNS, it then locates name servers for these hosts' domains which are queried using the Simple Mail Transfer Protocol (SMTP) to locate any e-mail information about the specified user. If a query is successful, the system attempts to find additional information using the *finger* protocol.

X.500 - The X.500 directory service is a standard service defined by the CCITT and ISO. It offers a hierarchical directory of services distributed across various X.500 servers which have authority over their portions of the hierarchy and may in turn delegate authority for sections of their own hierarchy.

It can be seen that these systems offer specialist services whereas systems such as WAIS and Gopher are designed for general purpose utility. Although they may be used individually, they become very powerful when integrated into general services such as Gopher. For instance, the Gopher server at Imperial College in London offers a complete section on directory services and the user may choose from a selection of services, including X.500 and Netfind, to locate the information required without needing to know how to access each of the offered services directly.

4.4. Hypermedia Systems with Restricted Distribution

Relatively few of the hypermedia systems that have been developed provide specific support for a distributed system. In fact, the most common systems are wholly based on single user systems (for example: Hypercard (Apple, 1987), Guide (OWL, 1987) and Storyspace (Eastgate Systems, 1991)).

Although there have been systems that are able to operate in a distributed environment, the type of distribution available in such systems can typically be divided into two classes: the incorporation of facilities that are available simply as a service of the host environment (e.g. the use of NFS in a UNIX environment), and utilisation of a central database (often based on a commercial database system) which client systems may use as a hypermedia server. The following sections describe some of these systems.

4.4.1. HAM/Neptune

The Neptune system (Deslisle and Schwartz, 1986) is a hypertext system designed to support computer-aided design (CAD) applications. It is based on a general purpose hypertext engine known as the Hypertext Abstract Machine or HAM (Campbell and Goodman, 1988). This provides a set of basic services for the storage and manipulation of nodes and links without actually applying any semantic knowledge to the data. This information may then be accessed by applications such as Neptune, which provide specialised presentation to the raw information stored in the HAM.

This system is an example of the centralised model of distributed hypermedia where a central server is used to store and manage all link and node information, which is then accessed by a number of local workstations. This distributed model was predominant in the early generation of distributed hypermedia systems, from the early examples such as ZOG and Augment, to more recent developments such as the HAM.

4.4.2. Knowledge Management System

The Knowledge Management System (Akscyn et al., 1988) is a commercial hypermedia system based on the ZOG system developed at Carnegie Mellon University in the seventies and early eighties. Although a powerful system, it now seems dated due to its frame-based interface and architecture. Although it cannot be considered to be an example of an open system, the version of KMS described in (Akscyn et al., 1988) is designed to be used in a networked environment, and is therefore an early example of a distributed hypermedia system.

It is a sign of the speed with which personal computer technology has advanced that the vision of a computing environment for the nineties at which KMS was aimed consisted of large scale networks of diskless workstations. To deal with such an environment, KMS is designed to provide a *master* server which stores details of the other storage devices containing KMS frames. Workstations may use Sun's Network File System to access these servers.

In fact, in a typical networked environment it is now usual for a user to have a personal workstation containing many megabytes of memory and a hard disk with a capacity of hundreds of megabytes. The users of such systems will typically all have large amounts of personal data, stored on their own local storage devices.

The result of this is that the distributed model provided by KMS becomes overstretched since each system is a potential source of frames. The master server must be able to store details about each of these systems, and offer this information to all clients. Clearly, the performance of this model would be compromised in such an environment. KMS is not flexible enough to operate in a peerless environment such as this.

Similarly, KMS was designed to operate within relatively small networks such as those operated within small companies. With the advent of world-wide networks such as the Internet, much greater connectivity of systems is taking place and the model provided by KMS cannot scale to encompass a network of this size.

4.4.3. Intermedia

A frequently cited example of a distributed hypermedia system is Intermedia (Haan et al., 1992). The general architecture of Intermedia is described in chapter three. The system is designed in such a way that it can only operate in a distributed environment, since all Intermedia workstations on a particular network expect to access a central hypermedia database. The system is limited however, since a workstation may only access a single server, and there is no mechanism by which servers may interact to provide additional information to their respective clients. This means that an Intermedia system is effectively restricted to a local area network containing a single server, with no mechanism for scaling the system to a wider area.

4.4.4. HyperForm

The HyperForm system (Wiil and Leggett, 1992) is designed to address shortcomings in systems such as HAM and Hyperbase where the hypermedia data model is strictly defined and cannot be modified as system requirements evolve. To this end, HyperForm incorporates an object-oriented database, which can easily be extended with new functionality. The flexibility of the system has been demonstrated by using it to simulate the functionality of various 'hyperbase' systems, including the HAM.

However, although the underlying storage model of the system is designed in a flexible manner, its approach to the distribution of information is similar to that taken by the systems already described, where a central server provides hypermedia functionality to client systems, which are responsible only for the presentation of information to the user. It does not specifically address the way in which a variety of networked systems may interact to supplement their respective facilities.

4.4.5. Virtual Notebook System

The Virtual Notebook System (Shipman et al., 1989; Burger et al., 1991), or VNS, previously described in section 3.5 is a hypermedia system designed to allow information acquisition and sharing, in particular for scientific communities. As such, distributed access to the information available is an important aspect of its functionality.

Initial prototypes of the system used Notecards and KMS, but Notecards did not allow groups of users to access a particular hypertext network at the same time, and KMS was unable to easily incorporate external information sources. Therefore a new system was developed, based on the relational database SyBase as a hypermedia engine, with specially developed presentation components running under X-Windows.

The distributed model of the VNS is based on workgroups, which each have a workgroup server (WGS). This server provides storage and management of local information through the facilities of SyBase. The WGS can also act as a gateway to other VNS workgroups, or to external information services. Although it is still based upon a central server, the VNS system can be seen to be more flexible, since it allows cooperation between distinct workgroups and sharing of information. The use of the SyBase system also brings advantages in fault tolerance, security and so on. Additionally the system is further enhanced by the ability to incorporate other information systems via 'gateways' in the server.

4.4.6. Sun's Link Service

Sun's Link Service (Pearl, 1989), also described in the previous chapter, is probably the best known example of an open, distributed hypermedia system. It represents a simple abstraction of links from documents which allows the hypermedia structure to be stored in a separate distributed database. Applications that wish to provide a hypermedia interface may then integrate with this link service using an open communication protocol.

Although the Link Service is undoubtedly an important example of the advantages of open hypermedia systems, it is lacking in many aspects of its functionality. In particular, Pearl does not discuss the distribution of link information beyond a single server, and unlike the VNS system described above, there is no defined protocol by which different servers are able to communicate in order to share information.

4.4.7. Summary

The discussion above shows that although there are distributed hypermedia systems in existence, they tend to offer a strict client/server interface where all hypermedia storage is provided by a central server, and the presentation of information is provided by the client system. In general, the systems are designed to provide distributed access to hypermedia information on a small scale so that a system based on a single central server is a feasible approach. With the notable exception of the VNS, these systems fail to take into account the possibility of multiple users who each have access to powerful workstations, any of which is suitable for use as a hypermedia server. This type of design is representative of the type of multi-user systems that were prevalent during the development stages of these hypermedia applications. Nowadays however, networks are more likely to be made up of various powerful workstations and personal computers, most with some form of local storage, and with key machines nominated as servers which have large scale storage facilities available. Systems such as those described above do not make full use of the distributed processing facilities of such networks.

Another drawback suffered by these systems is the lack of attention that they pay to the extension of the system beyond a local server. Only VNS offers the concept of multiple servers and links which bridge the information stored within them. This leads to the formation of 'information islands': numerous local systems which are unable to interoperate in order to enhance the quality of service provided with supplementary information from other systems. This is certainly not the type of system envisaged by Nelson, who saw hypermedia systems as a means to provide wide-scale dissemination of electronic information.

In addition to these problems, the use of central databases by these systems to store available information may be considered to be analogous to the use of proprietary formats by single user closed hypermedia systems such as Hypercard. By moving all information into the database, manipulating it using standard tools becomes difficult. In addition, these systems have little scope for the incorporation of links to other hypermedia systems, either of the same type, or through some interchange protocol, to a different system.

Finally, the scope for extending the functionality of the systems described is rather limited. Although many of the databases are designed to provide general purpose hypermedia storage mechanisms, they are strongly biased toward the link and node model of hypermedia and the incorporation of new functions requires a great deal of programming effort with wide-ranging effects across the whole system.

A notable exception is Sun's Link Service, which is specifically designed to allow a variety of independent applications gain access to hypermedia functionality. It does not however provide similar extensibility to the underlying Link Service. HyperForm is a more recent system which offers an extensible underlying database system, as well as an open interface to applications, but it is still oriented towards a strict client/server model.

Because of these disadvantages, the systems which have been described fail to meet the majority of the requirements for open hypermedia systems that were identified in chapter two. As well as the examples discussed above, many newly developed systems exhibit similar limitations in their designs (e.g. Hypernet (Marovac and Osburn, 1992), HyperArchi (Gaviotis et al., 1992), and Distributed Graph Storage (Shackelford et al., 1993)).

4.5. Truly Distributed Hypermedia Systems

Although the examples of distributed hypermedia systems described in the previous section can be seen to suffer from numerous limitations, recent years have seen new systems which offer more ambitious functionality. With the rapid increase in size of the Internet, and the equally rapid increase in workstation computing power, truly distributed systems have become a viable possibility. This section describes notable examples of these systems.

4.5.1. World Wide Web

The World-Wide Web (WWW) system (Berners-Lee et al., 1992), designed at CERN in Switzerland, began to be developed at around the same time as distributed information services such as WAIS and Gopher (see section 4.3). It uses a similar distributed architecture of multiple servers and heterogeneous clients, but relies on hypermedia links for navigation between documents and servers, whereas WAIS and Gopher provide query-based and hierarchical infrastructures, respectively. The system was originally designed to provide an information network to the High Energy Physics community, but its general applicability soon became apparent, and a wide range of information may now be accessed.

Unlike most of the systems described earlier, and as its name suggests, WWW was explicitly designed to allow a wide distribution of the available information across multiple servers. The basic type of information that is available through a WWW server is stored in the Hypertext Markup Language, or HTML (Berners-Lee, 1993b), format. This is a mark-up format based on SGML which is oriented towards the description of links in documents. Clients are able to access servers via a communication protocol known as the HyperText Transfer Protocol (or HTTP) (Berners-Lee, 1993a), in order to obtain HTML documents. The client must parse the HTML structure, and present the information and links that are embedded within it to the user. A WWW link can point to any document in any valid server. The HTML format also allows non-textual information, such as bitmap graphics, to be included in a document.

As well as supporting the transfer of HTML documents through the HTTP protocol, other well known access protocols, such as File Transfer Protocol (FTP), Network News (NNTP), Gopher and WAIS, can be used to incorporate additional information. The appropriate access type for a document is stored as part of the document identifier or Universal Resource Locator (URL). This identifier describes the access type, server and document to be retrieved and is of the following format :-

<access type>://<internet host address>[:port number]/<document path>

For example the following URL identifies an HTML document available from the CERN WWW server :-

http://info.cern.ch/hypertext/DataSources/Overview.html

The path component of the URL is formatted appropriately for the type of service being accessed. For example, it may contain a search string for services such as WAIS. WWW links specify a URL as the destination node, and therefore links to any information type are supported transparently.

Alternative information systems may be accessed in two ways. It is possible for a WWW server to offer a gateway to other server types which accesses the information, translates it into an HTML representation and supplies it to clients as if it was a normal HTML document. Alternatively, the appropriate functionality for common modes of access (e.g. FTP, Gopher) is typically built into the client applications which allows a direct (and therefore more efficient) connection to the appropriate service directly from the user's own system.

The functionality offered by WWW meets some of the requirements for open hypermedia systems, notably the ability to access a wide variety of distributed information sources. The flexible nature of the URL specification allows new information types to be incorporated very easily, and careful design of client and server systems would allow any additional functionality required to be added relatively simply. The nature of this extensibility also allows the system to potentially support various interchange standards; although it currently does not support other hypertext formats, these could be treated in exactly the same way as Gopher, for instance.

However, these advantages are concerned with the infrastructure of the supporting system, and are offset by the conflict with those requirements that are concerned with the system's approach to the hypermedia information it stores. The use of the HTML format and its embedded links mean that the actual information content cannot be manipulated independently using standard tools, and alternative link structures cannot easily be provided without duplication of the information stored, although the text-based nature of the HTML format is not totally restrictive.

This reliance on embedded links makes the incorporation of other forms of linking, such as information retrieval and string search, very difficult. Additionally the nature of the HTML format makes a clear distinction between the user and author of hypermedia information. In order to add links to HTML documents, the document must have additional mark-up added and, since there is no facility to update documents via HTTP, this must be carried out independently using a special editor, or by entering markup by hand.

It can be seen that although its approach to widespread distribution of information and access to alternative sources is a significant advance over many other systems, the rather basic approach to the provision of hypermedia structure is a limitation of the World Wide Web approach.

4.5.2. Hyper-G

The Hyper-G system, described previously in chapter three, offers a similarly ambitious approach to the distribution of information as the World-Wide Web, but also offers a distributed model (Kappe et al., 1993) that provides greater flexibility for the provision of hypermedia functionality.

Unlike WWW, Hyper-G separates link information from documents into a link database. This database is then able to act as a hypermedia server to various local clients. To retrieve a document, the client sends a request to the link server for the appropriate document object by specifying the object's unique identifier, the server returns access details for the document, and any anchors that exist in the document. The client may then use these access details to retrieve the actual document, and display it. This process is repeated as the user follows links and the client accesses the link server to access further documents. Because links are separated in this way, it would also be possible to overlay links on information held in other information systems.

In order to extend this model to allow wider applicability, a server identifier can be used in conjunction with the object identifier to allow objects to be stored by any server system. Each link is represented by a relationship between document anchors in the linked documents. In conjunction with this distributed link service, the document access provided by the system can be modified to include a cache system, speeding repeated access to distributed documents. The resulting distributed model is shown in figure 4.3.

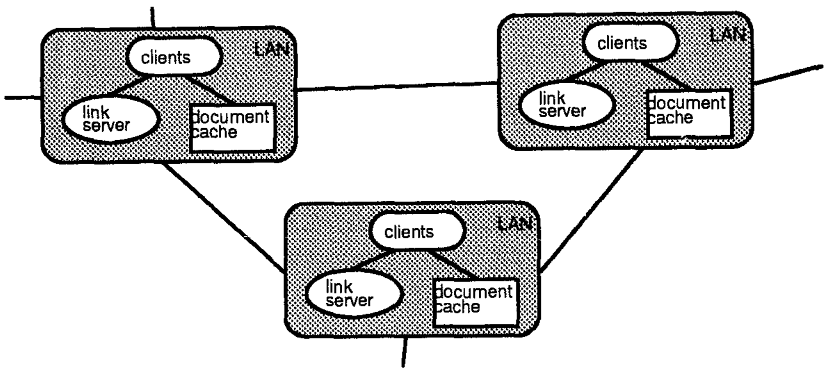


Figure 4.3 : Distributed Hypermedia Architecture proposed for Hyper-G (reproduced from (Kappe et al., 1993))

This model is currently under development, with the Link Server component based on the Hyper-G functionality described in chapter three. Thus the system can be expected to provide support for features such as automatic linking and multi-lingual access. The system however still suffers from the limited scope for extensibility that is inherent in the Hyper-G system.

4.5.3. DHT

The DHT (Distributed HyperText) system (Noll and Scacchi, 1991) is designed as a framework through which a variety of information services may be integrated. As such it does not provide any hypermedia storage or service, but simply consists of various server processes, which implement an open protocol allowing client systems to access them. These server processes exist simply to provide this consistent interface to a wide range of existing services such as relational databases, information servers such as those described in section 4.3, or to specially developed storage systems.

This approach has the advantage of providing consistent access for client applications to any integrated service. Since the information does not need to be translated into a special form for the hypermedia system to process, it is still accessible in its native environment; however, the server processes must be individually configured for each service that is to be integrated. This approach means that extending the hypermedia functions offered by the model, which are provided by the client/server interface protocol, requires a great deal of duplicated functionality to be updated.

4.5.4. Summary

The systems described above can be seen to offer a more flexible approach to the distribution of information, but they still suffer from certain problems that prevent them meeting the desired requirements of an open hypermedia system.

The World-Wide Web, for instance, requires its own documents to be structured using HTML, and encodes links within this format, although other access methods can be supported through 'gateway' modules. Thus, the information accessible through 'the web' is of a fairly static nature and alternative structures cannot easily be presented. In addition, extension of the system, for example the incorporation of new access methods, takes place in an ad hoc manner and there is no generic framework which allows this to take place in a seamless way.

The Hyper-G system offers some advantages over the WWW approach, since it separates links and documents. It is also able to incorporate external information services through the use of modules in the document caching system as an abstraction from client processes, and the separate link storage allows links within these systems to be made. Again however, no generic framework to support this form of extension is provided. Similarly, there is no general form of extension incorporated in the core Hyper-G hypermedia functionality. In common with WWW, the Hyper-G system is oriented towards presentation of the linked information. Neither of these systems offers a means of acting as an underlying link service.

The main advantage of the Hyper-G distributed model over the systems described in the previous section, is the degree to which a system comprising many servers is accommodated. Like the Virtual Notebook System, a local server may act as a gateway to other servers on a network on behalf of the local client systems. Because links are stored separately to the information which is linked, it is possible to provide alternative link structures very easily. However, there is little scope in the design for the provision of any form of linking other than the simple point to point link, or for the provision of hypermedia functionality to existing applications. This is a disadvantage, since a rich variety of alternative linking methodologies have been proposed by various researchers, which would enhance the functionality of the system should they be incorporated.

The DHT system offers a similar approach to that of the Hyper-G distributed model, in that it provides a number of servers in a distributed environment. The DHT approach however, stresses the use of existing information services via gateway servers which translate between the DHT protocol and the protocol of the particular information service being integrated. This provides an extremely consistent and integrated set of autonomous services which can easily operate in a heterogeneous environment. However, although a wide range of services may be integrated, the information they contain is translated into a basic hypermedia node and link model. The result of this is that specialised functionality within a service may be lost when accessed via DHT since it cannot be incorporated within the simple hypermedia model supported. Also, due to the discrete nature of the various server processes, extending the functionality of the DHT model requires a great deal of effort.

4.6. Conclusions

It can be seen that the distributed hypermedia systems that have currently been developed are not flexible enough to meet the vision of a truly open hypermedia system described in chapter two, or to approach the ideal of an 'information universe' envisaged by Nelson. Currently the widespread use of information servers, such as Gopher and WAIS, by information technology users with appropriate network access comes closest to providing the widespread, electronic dissemination of information anticipated by many authors.

Of the earlier generation of systems described in section 4.4, only VNS shows any of the flexibility that is required of a true open system. Systems such as HAM/Neptune, HyperBase and Intermedia utilise a central database for all objects, which results in a situation analogous with the proprietary formats of single user systems. Only VNS addresses seriously the approach to providing widespread distribution of information, but even the approach taken by this system is fixed, with little real scope for alternative linking models.

The HyperForm system is a more recent development than these systems, and attempts to address the issue of extensible functionality. It is successful in this to a certain extent, but while HyperForm provides a basic object-oriented framework which allows custom functions to be quickly incorporated into the core function set, it does not provide a framework for dynamically varying functionality. Like the other systems in section 4.4 it does not provide a flexible or extensible model for the distribution of HyperForm servers.

The systems described in section 4.5 typically offer much better support for the integration of additional information services. In particular, the widespread adoption of the World-Wide Web has lead to its classification amongst such systems. The whole architecture of the DHT system is oriented towards creating a common hypermedia interface to existing systems such as these. The most ambitious approach is seen in the design of the distributed model of the Hyper-G system, where a wide distribution of servers similar to that existing for the World-Wide Web community is anticipated, but with greater hypermedia functionality and stronger support for local client systems.

All of the systems described in this chapter still fall short of true open hypermedia functionality. Although they all meet some of the requirements that have been identified, they all fail to provide a true, dynamically extensible framework for the provision of hypermedia functionality, and this inflexibility is similarly mirrored in the tightly restrained distributed models employed by most of the systems. Just as a fixed set of hypermedia functions cannot be expected to meet the requirements of every user, it is short-sighted to expect a restricted distributed model to reflect all possible requirements of a network-based hypermedia system.

The following chapters in this thesis describe the design and implementation of Microcosm, the open hypermedia system developed in the Department of Electronics and Computer Science at the University of Southampton. Microcosm is designed to allow the requirements outlined in chapter two to be provided in one hypermedia system.

5. The Microcosm Framework

This chapter describes the aims and objectives of Microcosm, the open hypermedia system developed in the Department of Electronics and Computer Science at the University of Southampton. The following sections describe the key aims of the system, the history of the project and the design and implementation of the current version of Microcosm.

5.1. The Aims of Microcosm

The initial aim of the Microcosm project was to produce an open hypermedia system that could be used as a research framework within which the requirements outlined in the previous chapters could be investigated. As development of the system has progressed, it has begun to be used by other groups (for example, the History Department at the University of Southampton) as a flexible hypermedia solution. This also helps to evaluate the research work that is carried out.

The final system envisaged would be able to act as an underlying service to the user's whole environment, allowing all applications to support hypermedia links. The result would be an environment where all forms of information can be easily cross-referenced and integrated. As described in chapters two, three and four, this is not possible using most current hypermedia systems due to the restrictions imposed by the use of proprietary formats, and the lack of extensibility in both the media types supported, and the hypermedia functionality provided.

When gathering the requirements for Microcosm, a set of system principles were devised on which the design of the system could be based and which would act as a guide to the development of the system. These are summarised below:

- *No ultimate distinction between author and user.* All users should be able to link extra information into the system. This enables the user to create their own information structures, or to add links from shared information to additional personal information or annotations. These personal additions should not be available to others unless the user chooses to make them available.

- *Loosely coupled system with low interdependency between component parts.* By structuring the system so that the interface between components makes as few demands as possible, it is easier to incorporate other applications into the system. It also aids in meeting the third principle, modular functionality.
- *Modular functionality.* As the system is to be used in a research environment, it is important to allow all parts of the system to be replaced by experimental alternatives. This modularity allows the maximum flexibility for the system to be extended in order to investigate new ideas and techniques with the minimum of effort.
- *Separation of linking details from information.* The raw information (text, graphics, video, audio, etc.) is treated separately to the link details which define the relationships within this information. This brings numerous advantages: information can still be manipulated using standard tools as it has not been converted into some proprietary format, information that is stored in a read only form (e.g. CD-ROM) can still be incorporated, and alternative linking structures can be created to provide different views of the same information. These advantages were considered to outweigh the additional problems encountered in managing two separate bodies of information.

These aims are discussed in more detail in (Fountain et al., 1990).

5.2. Design History

The first stage in the development of Microcosm was a prototype system developed by Ian Heath for the Microsoft Windows 2.0 environment in 1989/90. When version 3.0 of Windows became available in the summer of 1990, the prototype system was ported to the new environment to take advantage of the increased stability and additional features provided. The prototype consisted of separate processes that were combined using the MS-Windows Dynamic Data Exchange (DDE) system for inter-process communication. The conceptual model of the prototype system can be seen in figure 5.1, and a full description of the implementation of the prototype system is given in (Heath, 1992). The system allowed a variety of discrete document 'viewers' to access an underlying link service, which provided a simple link database.

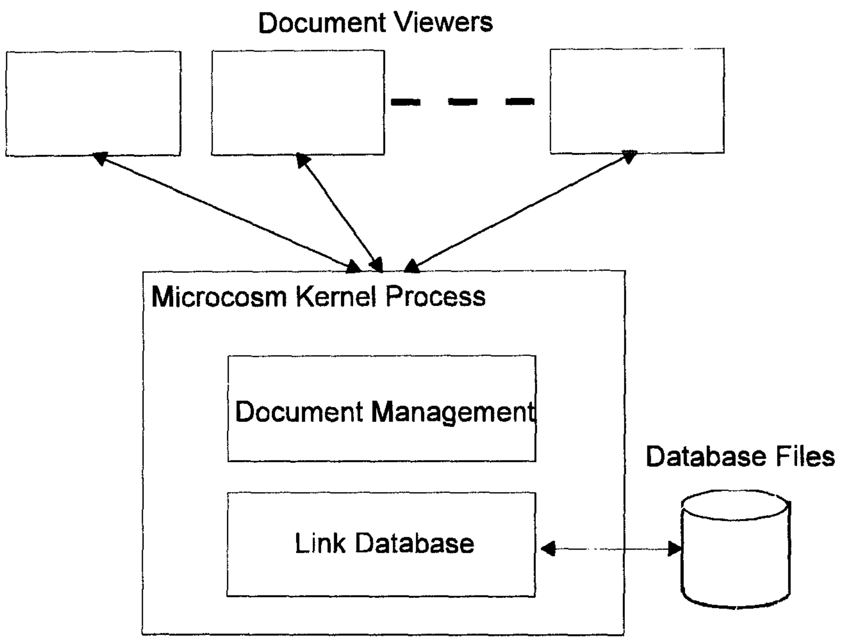


Figure 5.1: The Architecture of the Microcosm Version 1.0 Prototype

The prototype was used to demonstrate the feasibility of separating hypermedia links from documents into a separate link database, and the concept of a 'generic' link (Fountain et al., 1990). There was no concept of an 'author mode'; all users of the system were able to access the same linking and navigation facilities. Once the feasibility of these design principles had been established with this initial implementation, an enhanced system model was designed to support the aims described in section 5.1 that were not fully addressed by the prototype system. The main feature of this enhanced system was the provision of support for extensible functionality. Although the prototype consisted of separate processes, these were tightly bound together with a strictly constrained interface and could not be easily replaced by alternative processes. The new model is known as Microcosm version 2.0.

The main distinguishing feature of the new model is the separation of the system's functionality into a two layer system. These layers are: the front end of the system which provides control over the presentation of information and the user's interactions with this information, and the back end of the system that provides the underlying hypermedia linking service to the available information. The majority of the functionality of Microcosm is provided by this back end layer; its functions include the provision of linking facilities, presentation of links to the user, navigation tools, etc.

In the model, these layers are comprised of several processes. Each layer consists of a core control process which co-ordinates the behaviour of a number of child processes. These child processes each offer different elements of the overall system functionality.

The front end layer's control process is known as the Document Control System (DCS). It manages the execution of, and communication with, the various document viewers for different types of information. If new types of information need to be supported, additional viewers can be easily added since they are separate Windows programs.

The back end layer is managed by the Filter Management System (FMS). The task of this process is to control communication with the processes that provide linking functionality to the system. These processes, known as filters, must deal with messages from the front end system which request various actions from the underlying services available.

As a user interacts with the system, the DCS and FMS processes are continually exchanging information. For example, as users attempt to follow links, messages are sent to the FMS from the DCS. The FMS must then route the message to the filter processes to allow them to find any relevant links. If links are found, and chosen by the user, messages are returned to the DCS describing the documents that are the destination of the links. The DCS is then responsible for the presentation of these documents to the user.

The logical model for Microcosm 2.0 is illustrated in figure 5.2

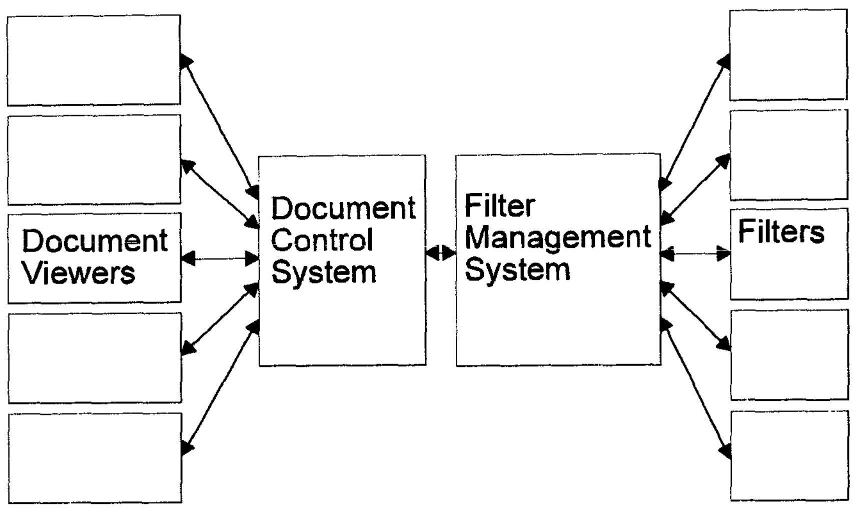


Figure 5.2 : Microcosm 2.0 System Model

5.3. Development of Microcosm 2.0

Having designed the conceptual system model of Microcosm 2.0, the various elements of the model were implemented for the Windows 3.0 environment. The choice of development tools was limited due to the status of the new environment. The system has been developed in C using the software development libraries provided by Microsoft. The following sections describe the implementation of the various system components.

5.3.1. Loosely Coupled Independent Processes

The first step in the implementation process was to develop the services required to build a system of loosely coupled processes. This requires two main services, a well defined message format to structure the communication between the component processes, and an underlying communication system to allow these messages to be passed between them.

The logical way to provide these services was as a set of libraries to which all parts of the system could be linked. The Windows environment is ideally suited to this approach since it supports the concept of a Dynamic Link Library (or DLL). These are shared libraries, and when an application is built using them, rather than bind all the library code into the application, it simply includes references to the library. When an application which uses DLLs is run, the Windows system loads all the necessary libraries into memory automatically. This has two significant advantages: if several applications use the same library, then the library is only loaded once, reducing the amount of memory used; additionally the specific implementation of such libraries can be transparently updated, without changing the dependent applications, simply by maintaining a consistent interface to the library. This is similar to the shared library concept supported by contemporary operating systems such as UNIX.

These libraries allow the Microcosm communication services described below to be implemented in an extremely efficient manner. The two main advantages are the memory saved by sharing the library between the many Microcosm processes, and the separation of the library code from the applications which use it, allowing straightforward maintenance of the underlying libraries.

5.3.1.1. Flexible Message Format

Because the system is designed to allow additional filters and viewers (both of which may offer any form of functionality) to be added easily, it is not desirable to define the message format used for inter-process communication too strictly. The prototype system used a small set of pre-defined fields for its messages, and therefore a new, flexible format was needed that would allow arbitrary new fields to be incorporated in messages should any part of the system wish to add more information to them. The size and content of messages had to be completely dynamic.

To achieve this, a free-format message structure was devised which encoded information about the actual fields in the message as well as the value of these fields. This is different to static message types which define a fixed set of fields as particular sections of a fixed size data block. Fields in the Microcosm message format are stored as couplets of field tags and field values. To create a whole message, these fields are simply concatenated together. Individual fields may be of any size.

The current implementation of this message structure is based on a simple plain text encoding which is both easy to implement, and easy to interpret. This latter feature is particularly important when debugging new system components or attempting to evaluate research results. In this implementation, field tags in the message always begin with a back-slash character ('\') and end with a space. The tag is then followed immediately by the field value. Any back-slash characters in the stored field value are identified by an additional back-slash as an 'escape' character; thus an MS-DOS filename such as 'c:\readme.txt' would be encoded as 'c:\\readme.txt'. A typical message is shown in figure 5.3.

```
\Action CREATELINK
\DocName C:\\MCMDOCS\\INTRO.TEXT
\DocType TEXT
\Description How to Build Microcosm Applications
\Selection Microcosm Applications
\Offset 143
\Date 30/10/91
```

Figure 5.3: A Typical Microcosm Message (formatted for clarity).

Function	**Purpose**
NewMessage().	Returns a new message pointer, which is initially empty.
AddTag(Msg, FieldId FieldValue).	Adds a new field to a message and returns an updated message pointer, or a null pointer if the message could not be updated. If the specified field already exists in the message, its value is updated by this function.
FindTag(Msg, FieldId, Buffer, BufferSize).	Fills the specified buffer with the value of the specified field, up to the buffer size. The return value indicates the number of characters copied to the buffer.
DeleteTag(Msg, FieldId).	Deletes a field from the specified message. The return value of the function is an updated message pointer.
DeleteMessage(Msg).	Frees the memory occupied by the specified message and renders the message pointer invalid.

Table 5.1: Functions provided by the Microcosm Message Library

This message format is supported by the functions available in the message library. This is a set of functions that allows a Microcosm process to build and manipulate Microcosm messages. An initial call to the library returns a pointer to a blank message. Other function calls may then be made specifying the message pointer, a tag, and a field value, to allow a Microcosm process to add, remove and modify fields or return the value of existing fields. These functions are described in table 5.1.

A number of standard tags are defined by Microcosm to provide the basic communication requirements, but any component process may include additional fields to extend the information that can be encapsulated in a message. These core tags and their meaning are given in table 5.2.

Field Tag	Field Value
Action	The action being requested by the message, for example "Open File", "Follow Link" or "Start Link"
DocName	The document on which the action should be carried out.
Selection	The information selected within the specified document upon which to carry out the required action
Offset	The position of the selection within the specified document.
DocType	The type of the specified document, to allow the appropriate viewer process to be used should the document need to be displayed.
Description	If the message specifies a link, this field is used to store the description of the link.

Table 5.2: Standard Fields of a Microcosm Message

An example of extended fields being used in messages is found in the link authoring process. To make a link, the link database (a Microcosm filter process) must receive a message with the 'Create Link' action. This message must specify both the start and end points of the link in question, and therefore requires two sets of DocName/Selection/Offset triples to be stored in the message. This is done by storing the fields for the link's start point as SourceDocName, SourceSelection, etc., and the end point fields as DestinationSelection, etc.

5.3.1.2. Inter-process Communication

The communication system must be able to transfer messages in the format described above (which may be of arbitrary size) between Microcosm processes. The communication system for Microcosm 2.0 was retained from the prototype system described earlier. This is based on the concept of an abstract channel which is established between two Microcosm processes. Each document viewer establishes such a channel to communicate with the DCS and, similarly, filters establish a channel in order to communicate with the FMS. Messages are passed between the front end and back end of the system by a channel that is established between the DCS and FMS processes. The connections between system components in figure 5.2 represent these communication channels.

When started, Microcosm child processes (filters and viewers) call a function in the communication library to establish a communication channel with the appropriate control process (FMS or DCS). As a result of this function call, the child process is issued with a unique channel identifier, and the control process to which the child process has connected receives a similar identifier, thus establishing a bi-directional link.

This identifier is used by a process (child or controller) when sending messages. It is passed to the communications library with a message, and is used by the library to route the message to the receiving process at the other end of the communication channel. The communication library delivers the message to the appropriate process in the form of a Windows message. The window which should receive these messages is identified by each connecting process when creating the channel. These channels are persistent for as long as a process wishes to remain connected to Microcosm; at any point it can terminate the connection by calling a function in the communications library.

The original communication library, used by the prototype system, utilised the Windows DDE system as its underlying protocol. However the DDE protocol is not well suited to the new unstructured message format described earlier. Additionally, its complicated operation makes it quite slow and thus created a significant performance problem in the new model with its greater reliance on message passing between component processes.

The DDE protocol is designed to provide an application independent interface at a high level and as a result is significantly more complex than is necessary for the low level inter-process communication required by Microcosm. Additionally, it is not designed to cope with the intensive levels of communication between many processes that comprise the Microcosm model. Therefore whilst the abstract channel interface was maintained, the underlying message system was refined and optimised by utilising the basic message passing model that is the basis of the

Windows architecture. This brought a considerable improvement in performance over the initial implementation of the library.

Using the communications and message libraries described in this section, the various components of Microcosm are able to work together as a complete system. When processes are changed or updated, the loose coupling provided by these libraries makes complex reconfiguration of the system unnecessary.

5.3.2. Document Control System

As described in section 5.2, the role of the DCS is to manage the document viewing services provided by Microcosm. Its main functions are outlined below:

- The DCS is responsible for starting viewer processes and initiating communication channels with them. When viewers are started, the DCS assigns them a unique identifier which they must add to any messages they send. This identifier is copied into any messages that are created as a response to the original message from the viewer. Thus if the DCS receives messages from the back end layer, 'in reply' to the initial message, it is able to route them to the appropriate viewer process.

- As users interact with viewers, the viewers generate Microcosm messages which are passed to the DCS. The DCS inspects these messages and, if appropriate, carries out some function; for example, the user is able to 'lock' a viewer process, preventing the DCS from instructing the viewer to display a new document. In most cases however, the DCS does not understand the messages it receives, and they are passed on to the FMS for processing by the back end layer of the system.

- The DCS also receives messages from the back end of the system via the FMS. As these messages are received, the DCS must decide if they are destined for a particular viewer process by looking for a valid viewer identifier, or take action itself. Typically the DCS is required to find a suitable viewer process to be used to view a new document. If a suitable viewer process is not available, a new viewer is started by the DCS.

- Viewer processes may be locked by the user, thus preventing the document they contain from being replaced by another. The locking process requires a message to be sent to the DCS, which maintains a record of which viewers are locked, to be used when the DCS is searching for an available viewer process.

5.3.3. Filter Management System

The most significant difference between the prototype system and Microcosm 2.0 is in the way that actual hypermedia functionality is provided. The prototype supported the use of separate viewer processes, and managed communication with them, and the front end of Microcosm 2.0 evolved naturally from the prototype. However, although the hypermedia functionality of the prototype was separated from the available information, this functionality was restricted to a single, simple link database that was bound into the kernel of the prototype system (see figure 5.1). The task of the back end layer of Microcosm 2.0 is to provide this separated hypermedia functionality in a more generalised and extensible manner.

As described in section 5.2, the back end layer of the system consists of a control process and child processes, organised similarly to the front end layer. For the back end layer, this control process is the Filter Management System (or FMS). The FMS delegates the task of providing hypermedia functions to a number of autonomous processes known as *filters*. These processes each offer a particular part of the overall hypermedia functionality. Thus it is very easy to extend the system, or to experiment with alternative implementations by adding or replacing filter processes. These processes are known as filters since their task is to filter out Microcosm messages that pass through the system and take appropriate action upon them.

A detailed examination of the development and evaluation of the FMS is found in chapter six.

5.3.4. Document Viewers and Filters

Document viewers and filters are the most important parts of Microcosm, since they provide all the 'real' functionality of the system, and are simply 'overseen' by their respective control processes. However, the role of viewers is more visible, providing the majority of user interaction with the system, whilst filters take a background role, often simply receiving messages from the FMS, replying with further messages, and not providing any form of direct user interaction. The roles of these two forms of process are described below.

5.3.4.1. Document Viewers

The role of the document viewer is to provide the user with the opportunity to view the information stored by the system and to investigate the links between documents. Typically, a viewer process uses the standard Microcosm communication services and message format to communicate with the DCS. Viewers using this approach must be specifically designed for use with Microcosm, or be able to be modified appropriately. However, it is not possible, or desirable, to require new viewers to be written for every possible document type just so that the Microcosm communications system may be used.

Ideally, any application should be able to be used as a viewer in some way. This requirement has led to the development of a wide range of support for viewers which are able to offer differing levels of Microcosm awareness. The different forms of viewer integration are described below.

- A *fully aware viewer* uses the standard Microcosm communication system as described above. It is able to both receive, understand and respond to Microcosm messages, and also generate such messages to send to Microcosm.

- *Partially aware viewers* are viewers where a level of configurability in the application allows some Microcosm functionality to be incorporated. These are typically third party applications such as Word for Windows or Toolbook. However, the degree to which such applications can be integrated depends on the flexibility built into them. For example, Toolbook is able to access functions in Windows DLLs, and thus is able to connect to Microcosm and send messages. Unfortunately the programmability of the system is oriented towards supporting user interaction and Microcosm is unable to deliver messages to it using the normal communication system. However, by using an additional process to mediate between Microcosm and Toolbook, messages can be delivered. Other applications such as Microsoft Word and Microsoft Excel support external communication through the Windows Dynamic Data Exchange (DDE), and again an intermediate process can be used to mediate between Microcosm and the external application using DDE.

This process simply converts Microcosm messages to a form understood by the third party application and delivers them using DDE. If necessary, the application can send DDE messages to the intermediate process which are then converted to the Microcosm format and delivered to the DCS.

The partially aware category covers a wide spectrum of integration, from powerful, extensible applications such as Microsoft Word which approach full awareness, to less flexible systems, such as Authorware, which offer much more restricted extensibility to the user.

- The third possibility is an *unaware viewer.* These are typically simple Windows applications such as the Notepad text editor, or the SoundRec wave form editor. Although there is no way to make these applications understand or receive Microcosm messages, they can still be used as viewers. To view a document, the application can easily be started with a filename specifying the document to be viewed as a command line argument. The method by which such applications are able to access the hypermedia functions provided by Microcosm is more subtle. Like most graphical user interfaces, Windows provides a clipboard facility for the transfer of data between applications, and almost all Windows applications support the use of this facility. Any application that registers its interest in the clipboard with Windows is subsequently informed when changes occur in the information stored there. By making Microcosm aware of the clipboard, many existing applications can be easily integrated into Microcosm without needing to make any changes to them. Simply making a selection, in the Notepad application for example, and copying this selection into the clipboard, causes Windows to notify the DCS that the clipboard has changed. This allows Microcosm to retrieve the clipboard contents, and create a Microcosm message to be sent to the back end layer. This message can be of any type, depending on the type of function the user wishes to access, for example 'Follow Link'.

This range of support allows Microcosm to provide an underlying hypermedia service to virtually all Windows applications, effectively combining those applications into an integrated information environment. The issues involved in integrating a wide range of applications with Microcosm are examined further in (Davis et al., 1992b).

5.3.4.2. Filters

As described earlier, a filter's relationship with the FMS is far more subordinate than the equivalent relationship between viewers and the DCS. A filter simply waits for messages to be sent to it from the FMS, and then takes some action. When it receives a message, a filter has several options. Depending on the contents of the message: the filter may choose to block it and prevent any other filters from accessing the message, alternatively it may carry out some processing and generate other messages, or the message may not be understood by the particular filter and is just passed back to the FMS. In some cases a filter will do both of these things, for example a filter creating links will receive and block 'Start Link' and 'End Link' messages and will subsequently generate a 'Create Link' message combining the information from these two messages.

The use of separate processes to provide the hypermedia service means that, as with document viewers, third party applications may be utilised by Microcosm as filters. However, to use such applications as filters, they must be capable of maintaining a two way dialogue with the FMS. This restricts the choice of applications to those which are fully Microcosm aware, or which offer some form of two way external communication. The latter type of application may be integrated through the use of an intermediate process between itself and the FMS in a similar manner to those used between the DCS and third party viewers. An example of this was the use of the Superbase for Windows database application as the main Microcosm link database during early development of the system. A Microcosm filter was created which would translate Microcosm messages into a form which Superbase could understand, and vice versa.

Chapter six provides a detailed discussion of the role of the filter system and describes some of the filters that have been developed.

5.3.5. Docuverse

Although initially Microcosm accessed documents using the basic MS-DOS file system, it was soon apparent that the limitations this approach created on the size of filenames, and the limited set of file attributes that MS-DOS provides were making life difficult for users of the system. Therefore, it was decided to introduce a document database (or *Docuverse*[1]) to Microcosm, even though this represented a compromise since we wished to keep the system as open as possible.

The problem created by introducing an additional layer between Microcosm and the file system is that only applications which are fully Microcosm aware, or which can access remote functions in Windows DLLs, are able to use the alternative file access functions that are provided. Also, since the Docuverse is not a true component of the underlying file system, it does not keep track of any changes made to files using the normal file system interface. For example, a file might be moved or deleted without the Docuverse being updated. However, there are advantages to the Docuverse, since users may now view meaningful document names, establish properties (such as document types, authors and subjects), and a query facility provides an alternative method for the location of appropriate documents. It was decided that the advantages outweighed the negative considerations and the Docuverse was implemented, and incorporated in all the necessary Microcosm components (i.e. DCS, viewers, filters which deal with document details). Its design and development is described in (Edwards, 1992).

5.3.6. System Configuration

In the preceding sections, the flexibility of configuration provided by the Microcosm 2.0 model has been clearly demonstrated. However, this flexibility must be supported by some form of configuration facility so that the two control processes are aware of the particular service processes that are available at any particular time, or in a particular context. This configuration information is provided by a set of Windows initialisation files.

Rather than fix a particular configuration with a single initialisation file, a hierarchical system was developed, based on the concept of a configuration that would be dependent on the particular system installation, the current user and the *application* chosen by this user. A Microcosm application is a concept used to identify the particular configuration requirements of a particular subject area. It identifies a subset of the files described in the Docuverse, and a configuration file that describes

[1] The term docuverse was first coined by Ted Nelson when describing the architecture of the Xanadu system.

any special aspects of the application. Information about the available applications is stored in the top-level configuration file. By combining the top-level configuration file, an application configuration file, and a user's own configuration file, the inherent flexibility of the Microcosm model is kept as open as possible.

General configuration information for a particular Microcosm installation can be stored in the top-level configuration file, and the details it stores apply whenever Microcosm is run. At the next level down from this, the configuration files for the applications available may specify the particular filters that are required to provide the underlying hypermedia support specific to each application, and any special viewers that are required for the application in question. Finally the user's own configuration file allows personal preferences to be included in the configuration. For example, personal link databases and navigation tools, or preferred viewer processes. This configuration hierarchy is illustrated in figure 5.4.

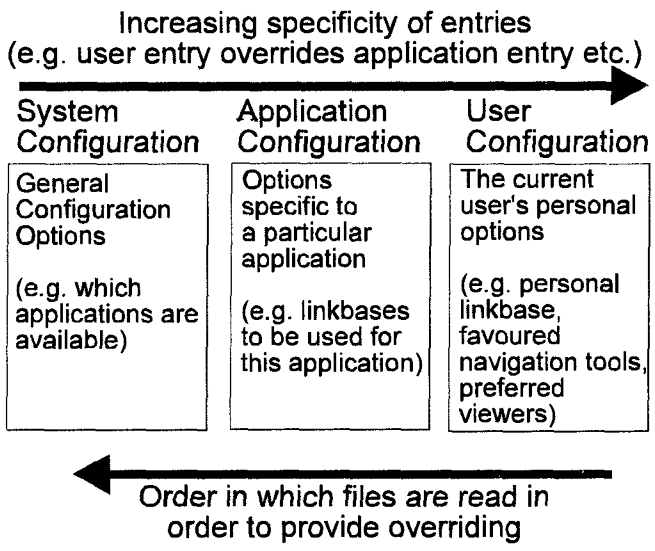

Increasing specificity of entries
(e.g. user entry overrides application entry etc.)
System Configuration
Application Configuration
User Configuration
General Configuration Options
(e.g. which applications are available)
Options specific to a particular application
(e.g. linkbases to be used for this application)
The current user's personal options
(e.g. personal linkbase, favoured navigation tools, preferred viewers)
Order in which files are read in order to provide overriding

Figure 5.4: The configuration file architecture used by Microcosm to vary system configuration according to the current user and the application which they are using

The type of information stored in this file hierarchy is mainly concerned with the service processes (viewers and filters) that are available. The DCS, for example, accesses a list of the document types understood by the system and details of the viewer processes that should be used to display these various types of document. Additional details are stored which let the DCS know whether the various viewers are Microcosm aware, and hence how to inform them of the document to be displayed once they are started. Similarly, the FMS may access the appropriate configuration files to establish which filter processes should be started.

It is possible that the information stored at the various levels of this hierarchy will overlap: for example the top-level file will normally define the viewer for text documents as the standard Microcosm Textview program, but a particular user may decide to use Microsoft Word for its powerful text processing facilities. In this case a priority system is used to establish which configuration file takes precedence; the entries in the application level override comparable entries at the top level and similarly the user level overrides the application level. This mode of operation is provided by a special DLL that Microcosm processes may use to access and update configuration files. Calls to read a particular entry are carried out by first looking in the user level and working upwards through the levels until a matching entry is found. When a user's interaction with the system requires new configuration details to be stored, this information is always written to the user level configuration file; changes to the upper levels have wider ranging implications and must be carried out as specific maintenance tasks. If specifically necessary, a Microcosm process may override the priority system by using the standard Windows support for configuration files. For example, the FMS requires details of all filters at each level and the overriding facilities would defeat its attempts to access these details. Therefore it uses the standard Windows functions to read this information from the appropriate files.

5.4. Microcosm in Action

Our work with Microcosm has resulted in the development of many applications, mainly for use within the university environment, but also including applications such as engineering quality control and computer aided manufacturing. Whereas the preceding sections of this chapter have described the system model and development of Microcosm, this section outlines two possible scenarios illustrating how hypermedia 'information sets' developed using Microcosm might be used.

5.4.1. Open Hypermedia as an Educational Resource

As a research group, we have undertaken a great deal of collaborative work with the History Department at Southampton who have utilised Microcosm to investigate the use of information technology in teaching History. This has included the development of a hypermedia system which contains documents related to a sequence of events that took place in Yugoslavia during the Second World War. This section describes how a student might use this system to investigate the historical significance of these events.

> When starting the system, the student logs in with a personal identifier and chooses the Yugoslavia application. This allows the correct set of filters to be started by the system (in particular the user's personal link database to store private links in), by identifying the various configuration files that should be used. Once Microcosm has started, the student can select documents to view using the DCS. Typically this might be the introduction document which details the major information resources available via the system. From these resources, the student chooses the 'question file' which outlines the historical points that are to be investigated by the use of the system, and sets appropriate questions for the student to answer. In the Yugoslavia application, question one asks the user to investigate the reasons why a planned sabotage operation to blow up a railway bridge did not take place.
>
> In the text of the question, the main Yugoslavian protagonists - Mihailovic and Tito - are named, and identified as leaders of opposing resistance groups. The student selects each name in turn, and is able to obtain background information by selecting these names and selecting the 'Follow Link' option from the viewer's menu. The links identified point the student to biographical information in a Microsoft Word document and digitised photographs of the two opposing leaders. From the biography, the student discovers more details about the two resistance groups (Partisans and Chetnicks) which were represented by the men, but attempting to follow links to additional information from these names proves unsuccessful.

Instead the student re-selects the text containing the names of the resistance groups, and carries out a 'Compute Link' operation. This operation is activated by a Microsoft Word Macro that has been added to Microsoft Word's menu structure for use with Microcosm. The macro generates a Microcosm message with the action 'Compute Link' which passes through the Microcosm framework to activate the Compute Links filter. This filter carries out a text retrieval operation based on the set of available documents, attempting to locate documents that are deemed to match the user's selection. In this instance, it successfully identifies several documents that contain elements of the selected text, including a document that discusses the evolution and motivation of the two resistance movements.

Having read this background material, the student uses the History filter, which has maintained a record of all the documents viewed in the session, to return to the introduction document in order to investigate the other sources of material available. These resources include transcriptions of government meetings, memos from British agents working with the Yugoslavian resistance, operational plans, digitised maps, diagrams, and recorded interviews with key individuals such as the British agents involved in the sabotage operation. These documents are all interlinked, using authored links (both generic links with dynamic anchors, and normal anchored links) and the dynamic text-retrieval links of the Compute Links filter.

The student may browse through these sources, using links to jump between relevant items and thus explore the various accounts of the events that took place. When satisfied with the understanding of the events that has been reached, the question file is once again retrieved and used to launch a Windows Notepad process in which to enter an answer to question one. When the answer is complete, it is saved and the student invokes the HiDES filter (see section 6.2). This filter analyses the answer and dynamically generates a suitable response. This response remarks on the points raised in the student's answer and may suggest other documents that the student should read or re-read for further understanding. Since the answer is presented simply as another document, these suggestions can be expressed as links, and the student is able to go directly to the appropriate documents.

Based on the response from the HiDES system, the student may choose to view further material and revise their answer, or may continue onto the next question. Used in this way, the system may act as an information source in the preparation of work, or alternatively the student could be instructed to add further commentary to the system and submit their additional material and links for assessment.

5.4.2. Open Hypermedia as the Basis of an Integrated Environment

This section describes an alternative and more ambitious scenario where Microcosm may be used by a typical computer user (perhaps a research student), as an integrating agent for their whole environment. In this scenario, Microcosm is not so much a prominent presentation system as an underlying, enabling technology.

Typically the user arrives in the morning, logs into a workstation and starts Microcosm. The first activity undertaken is starting a terminal session in which to check for new e-mail. One of the messages received is from a colleague who asks a question about a particular piece of software designed by the user. To compose a reply, the user selects the name of the software in the original message and attempts to follow links the e-mail program. This activates generic links from the phrase selected to the various documents and source files that comprise the piece of software in question. From this list of links, the design documentation for the software is selected and appropriate parts of the documentation are copied into the reply being composed.

Once the day's e-mail has been dealt with, the user opens a document currently being prepared which discusses the results of recent research work. The section of the document being written discusses the relative merits of similar work, and the user needs to locate suitable references for the discussion. By selecting the names of the systems in question from the document, a query is submitted to the Compute Links filter. The resulting text retrieval search identifies a list of the user's documents have content relevant to the systems in question. These documents are presented as a list of links from which the user may choose. Browsing through some of the documents found by the search, it is discovered that similar sections may be used as a basis for the section being written in the new document.

Rather than immediately copy this information into the current document, the user creates a link from the new document to the relevant sections in the existing documents. This link is created in the form of a button which will act as a reminder when the section finally comes to be written. The remaining sections of the document may then be planned in a similar way, searching out relevant information and linking in pertinent results of various types (e.g. spreadsheets, diagrams, etc.) before actually committing to a linear form of the document.

Moving on to other tasks, the user accesses resource discovery tools such as Gopher and World Wide Web (see chapter four) to search for information. Although the client programs for these systems do not have built in Microcosm awareness, they can be used as Microcosm unaware viewers as described in section 5.3.4.1. They are able to access Microcosm's hypermedia functionality by using the Windows clipboard as a communication method. In this instance, a link is made from the user's document space to a World Wide Web page. Although unaware viewers cannot be used to end a link directly, Microcosm allows a link to be completed by specifying the client process to be run and the command line arguments required to access the correct information. The particular page chosen reflects the current state of several projects in which the user has an interest, following the link in future will allow the user to access up to date information about these projects.

This scenario shows how the user is able to utilise Microcosm as an information management service whenever he or she accesses electronic information.

The two examples presented in this section illustrate the flexibility of the open framework which Microcosm represents. They also illustrate the type of interaction favoured by Microcosm, that of an exploratory interaction where the user first identifies an area they are interested in and then investigates the facilities provided by Microcosm in order to move on to related information. This is in sharp contrast to the directed interface found in the hypermedia information that can created with earlier generations of hypermedia systems which use buttons to highlight the available routes through the information as chosen by the author, rather than allowing the user to devise a path best suited to their needs.

5.5. Summary

This chapter has outlined the open approach to the provision of hypermedia functionality through the use of the Microcosm system. In particular this is provided by the modular nature of the design of Microcosm, and the open protocol which allows these various modules to communicate. This allows a variety of applications to be incorporated as viewers, and the underlying hypermedia services may be extended by adding filters to the system.

The open architecture meets the requirement for the system's functionality to be extended, and all interaction with the system takes place through the same interface with no distinction between users and authors. The use of separate link databases via the Filter Management System prevents marking up documents with anchor information. At present the system offers limited scope for access to distributed information, however the development of a distributed version of the system is described later in this thesis. Since there is no accepted standard for interchange of hypermedia information, there is no specific support for standards in the system at present.

Section 5.4. showed how the open architecture of Microcosm allows it to be applied in various ways to the task of information management. In particular, it illustrated how Microcosm is able to utilise existing tools as well as the facilities provided specifically as part of the system.

The following two chapters describe in detail the design and development of the Filter Management System which manages the back end layer of Microcosm, and the extensions made to the FMS to allow the development of a distributed version of Microcosm.

6. Filters and Filter Management

6.1. The Role of the FMS

As described in chapter five, the two core Microcosm processes (Document Control System (DCS) and FMS) do not provide hypermedia functionality themselves. The DCS simply starts up viewers as required and co-ordinates communication between the current set of active viewers and the back-end of the system. Messages from viewers are received by the DCS and passed to the FMS, which then guides these messages around the current set of filters. Similarly, messages that are returned from the filter set are sent back from the FMS to the DCS, which in turn passes the message on to the appropriate viewer process or performs some action itself.

The role of the FMS therefore is to act as a 'junction box', routing messages between the filters that are present in the system. It is this set of filters that defines the hypermedia functionality available to the user. By changing, adding, or removing filters, the functionality of the system can be tailored to suit the specific needs of a particular user, or Microcosm application.

The functions provided by the various filters can be invoked in two ways:

- By the receipt of a message which specifies an action that the receiving filter is able to carry out. The sending of such a message typically takes place when a user makes a selection in a document being viewed, and then chooses a function from the menu of a viewer process; for example, Follow Link or Start Link. This action causes the viewer process to create a Microcosm message which encapsulates details of the action chosen by the user, and the selection on which the action is to be carried out. This message then makes its way through the elements of the system as described earlier.

- Alternatively some filters have functions which may be invoked directly by interacting with the filter process. Typically, navigation-oriented filters, which monitor the user's progress and provide context-based functionality offer this type of direct access to the information they have gathered; for example the History filter described later in this chapter allows direct access to the set of documents that have been previously seen.

At any time when using the system, the user may choose to reconfigure the system to suit their current requirements. This is done by using the FMS to update the current filter set (the dialogue box shown in figure 6.1 shows how this facility was provided in one implementation of the FMS; the current list of filters may be re-ordered, or additional filters can be executed and added to the current list).

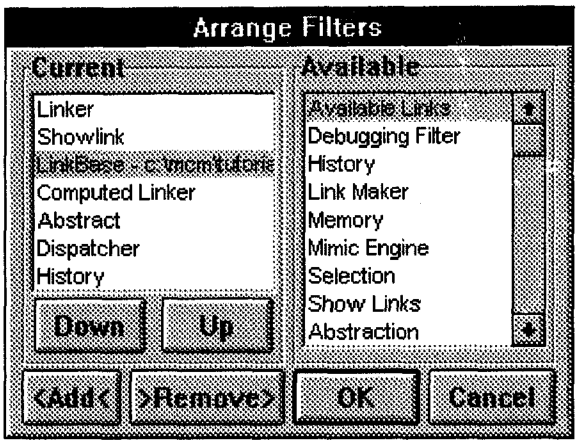


Figure 6.1: Filter Managemeni System dialogue box used to manipulate the configuration of filter processes in the first implementation of the FMS.

It is this total flexibility that is available in the configuration of the system, and the wide range of functions which filters are able to provide, that allows the services offered by Microcosm to develop and expand over time, continuously adapting to changing requirements. For example, filters may be used to provide navigation facilities (Hill et al., 1993), or as a way to add alternative linking paradigms such as a text retrieval system (Li et al., 1992). Similarly, as new viewer processes are added to the system to handle new media types, any special functionality required can easily be added to the system by creating new filters.

6.2. Building a Hypermedia System with Filters

With the presentation facilities for Microcosm provided by the front-end layer (DCS and document viewers), it is very easy for the back end layer to offer a simple hypermedia system with a small set of basic filter processes. The essential requirements of such a system are the ability to create links, access links and present these links to the user. The filters required to offer this functionality are described below.

Since Microcosm stores links separately from the documents being viewed, it is first necessary to have a filter providing a database of links, known as a 'Linkbase'. This is arguably the most important filter, since links or associations, in some form, are the most basic component of a hypermedia system. The primary function of the Linkbase is to process messages with the 'Follow Link' action. These messages contain a selection that the user wishes to try to find a link from, along with the source document, and offset within that document, of the selection. If a link is found in the link database that matches these criteria, a description of the link and a specification of its destination is sent back to the FMS in the form of a message with the action 'Dispatch'. The 'Dispatch' action indicates to the DCS that the document specified in the message should be presented to the user by being *dispatched* to a viewer process.

When they are received from the Linkbase by the FMS, these messages continue through the system, where they may be processed by other filters, and are eventually presented to the user. At any one time there may be a number of different Linkbases active within the system, each providing different 'views' of the relationships between the available documents.

As well as 'Follow Link' messages, the Linkbase also understands messages specifying other actions. Viewers are able to interrogate the database with a 'Find Buttons' message which causes the Linkbase to return messages which describe the locations of any buttons in the specified source document, allowing the viewer to highlight these buttons in its display. Also understood are messages with the 'Create Link' action; these describe new links which are to be added to the Linkbase. Because a Linkbase is simply one of many filters, and the user may employ a number of different Linkbases, a Linkbase which receives a 'Create Link' message does not send the message back to the FMS, thus avoiding the same link being added to all active link databases.

Typically, a user will use a personal Linkbase to record any new links made, but will also have additional Linkbases installed that provide links made by other authors. Full details of the development and implementation of the Microcosm Linkbase are given in (Heath, 1992).

The 'Create Link' messages described above are formed by a separate 'Linker' filter. This receives 'Start Link' and 'End Link' messages created as a result of user interaction with viewer processes, and allows the user to bind them together to provide a full link definition, with source and destination details, description, and type of link. This information is used to form a new message with the 'Create Link' action. Although this functionality could be included within the Linkbase itself, by keeping it separate it is possible to provide alternative ways to create new links (e.g. automatically as a result of document analysis) and a consistent link creation interface can be provided to different implementations of the Linkbase. The ability to make links can also be easily controlled by restricting user access to this Linker filter.

One further filter is required to complete the basic linking functionality of the system. This is the 'Available Links' filter which captures and displays the 'Dispatch' messages generated when links are followed. Without this filter, all the links found would be returned to the DCS and activated automatically. If a large number of links were found this would be both time consuming and confusing for the user since it would result in a chaotic array of viewer processes being started to display the newly found documents. The Available Links filter simply displays the descriptions of the links (from the 'Dispatch' messages) in a dialogue box, and only allows the messages that represent the links chosen by the user to be sent back to the DCS. Figure 6.2 shows these basic filters and illustrates the type of messages which they are able to process and generate.

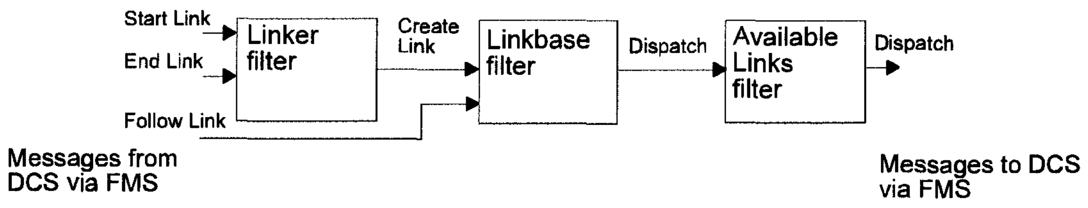


Figure 6.2: The Microcosm messages understood and generated by the basic filter processes which are required to provide simple hypermedia functionality.

The filters described so far offer only the most basic hypermedia facilities to users of the system. However, these are merely a basis around which a more sophisticated system may be created. This may be done by installing more filters to provide additional functionality, or by replacing existing filters with others that offer alternative facilities. The following list outlines some of the other filters that have been created to enhance the basic functionality of Microcosm.

- *Short Term Memory.* This filter is an alternative to the standard Available Links filter described previously. It implements a 'short-term memory' of the links that have been followed, and uses this to indicate to the user whether a link has already been chosen recently. This is based on the work described in (Briggs et al., 1993).

- *Abstract.* (Beitner, 1993). This filter provides another alternative to the Available Links filter, this time for visually complex documents (for example video sequences or raster graphics). Rather than attempting to describe the destination document of a link with a short piece of text, this filter presents the user with a small visual representation, or visual abstract of the destination document. This is able to impart more information to the user than a text description. For example, a video sequence can be represented by a short sequence from the full video clip, while bitmap and other graphic documents can be shown at a low resolution to give a rough guide to the content of the document.

- *Local Map* (Hill et al., 1993). This filter acts as a navigational aid by showing the links available from the user's current location in 'hyperspace'. It shows the user a spider diagram with the current document at its centre and the documents which are linked to it arranged around the edge of the diagram. The user is able to obtain the document's name and media type from the display provided, and can follow the links shown directly from the map if they so wish.

- *Compute Links* (Li et al., 1992). This filter offers an alternative mechanism to 'traditional', authored hypermedia links. Sets of documents are pre-indexed by a utility program which attempts to identify keywords in the document and builds a lookup table based on the statistics it collects. When the filter receives a message with the 'Compute Link' action, it attempts to select documents from its index that form an appropriate match with the selection in the message using a text-retrieval algorithm. The filter then creates 'Dispatch' messages describing the matching documents. This allows the documents found to be presented to the user in the Available Links filter's dialogue box, in the same way as links from the Linkbase filter.

- *HiDES* (Colson and Hall, 1991). This filter was developed in collaboration with the History Department at the University of Southampton. As described in chapter five, students are able to use a Microcosm system to investigate historical events. The students are set questions and can browse the available hypermedia network to investigate relevant historical material. They can then submit answers to the questions to the HiDES (Historical Document Expert System) filter. This filter analyses the answer and generates a suitable response. This response is then presented to the student, and may include links to further relevant documents that will allow the student to improve their answer to the question.

The above filters illustrate just some of the possible extensions that can be provided within the Microcosm framework. Other filters exist which aid development work by monitoring, displaying, and logging the messages that pass through the FMS, whilst many other possible filters have been suggested. These include a synonym generator which could be applied to selections to expand the number of links found in a sparsely linked information set and an intelligent filter which could build a context model of the user's requirements in order to restrict inappropriate links in a heavily linked set of documents, thus reducing 'link overload'. Some other possible filters are described in (Davis et al., 1992a).

The rest of this chapter investigates the way the FMS process has been implemented in order to support the functionality described above.

6.3. Simple Filter Management

6.3.1. Design of the FMS

The initial design of the FMS organised the set of active filters into two simple linear chains as shown in figure 6.3. This model was based on an interpretation of filter functionality that clearly identified two basic types of filter. The first type was a filter which would deal with link processing, responding to requests to find or create links: for example, the Linkbase filter described in the previous section. The second type of filter was concerned only with the manipulation of links found by the former filter type, links being identified as messages with action type 'Dispatch'. An example of this type of filter is the Available Links filter described earlier.

The use of a chain arrangement allows a filter to affect the rest of the currently active filters, depending on the way that it reacts to the messages it receives. For example, a link database will create additional messages describing links, based on the content of a 'Follow Link' message. Some filters may choose to capture a message and prevent it from being sent to other filters in the chain. An example of this is the processing of a 'Create Link' message by a Linkbase; rather than have the link added to all Linkbases that are installed, a Linkbase which receives a message of this type must not return it, but simply adds it to the link database file which it is using. This prevents the link from being added into the link databases maintained by other Linkbase filters in the chain.

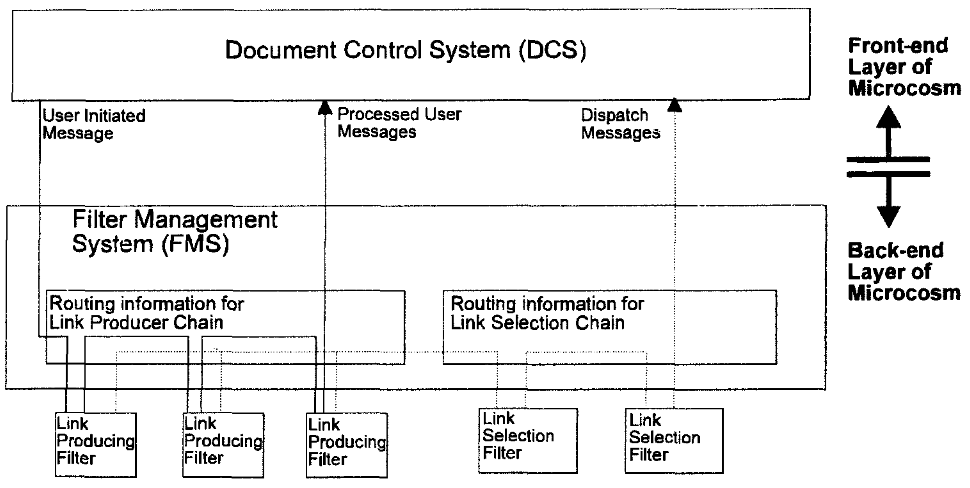

Document Control System (DCS)
User Initiated Message
Processed User Messages
Dispatch Messages
Filter Management System (FMS)
Routing information for Link Producer Chain
Routing information for Link Selection Chain
Link Producing Filter
Link Producing Filter
Link Producing Filter
Link Selection Filter
Link Selection Filter
Front-end Layer of Microcosm
Back-end Layer of Microcosm

Figure 6.3: The initial dual filter chain design of the simple Filter Management System

As this version of the FMS received messages from the DCS, it would send them first to the 'link producing' filter chain, initially to the first filter in the chain. This filter would then inspect the message, take any action necessary and return the message, followed by any additional messages that were created, to the FMS. The FMS would inspect messages returned by filters and any 'Dispatch' messages would be moved into the second filter chain. Other messages, such as 'Follow Link' messages that had been processed by the returning filter, were simply sent to the next filter in the chain so that they gradually progressed to the end of the chain, at which point they would simply be sent back to the DCS. Although the DCS currently simply discards messages from the FMS other than those of type 'Dispatch', this arrangement provided for possible future developments to the DCS.

When 'Dispatch' messages were diverted to the second filter chain, they made their way along that chain in exactly the same way as messages in the 'link producing' chain. The filters found in this chain however typically did not create other message types, they simply received 'Dispatch' messages and took action according to their particular function. This provided these filters with an opportunity to influence the way the user perceived the link structure of the information they were browsing. There were a range of different filters of this type, from the Available Links filter that lists the links found, to the History filter that simply monitors the links chosen by the user, in order to record the path taken through the available information. An example of the many other possibilities for the filters in this chain is an 'intelligent' filter that could attempt to intercept links that were not considered appropriate for the user's current needs.

6.3.2. Implementation and Refinement

The dual chain design described above was successfully implemented, but when it was put into use outside of the development environment, by users who were not familiar with the rationale behind the design, it soon became apparent that the dual filter chains were confusing to many users who had less insight into the way that the underlying message passing took place. This made it very difficult for these users to understand how to configure the system, since they were unable to decide which chain filters should be placed in. Another problem for users was the added complication of independently altering the filter configuration in each chain.

Therefore it was decided to simplify the design so that it treated all filters equally, placing them within a single, unified, chain. This modification was carried out, and proved to be very successful. Although theoretically the use of a single chain increases the need to be aware of how filters are ordered (for example, filters which would have been within the Link Selection chain in the initial model need to be located after any Linkbases in the single chain), it has the advantage of treating the whole configuration as one entity. This simplification, combined with a little intuitive understanding of the relationship between different filters, and the functions they provide, enables users to work successfully with various filter configurations with greater confidence.

This refined implementation of the FMS has seen a great deal of use by various groups within the University of Southampton, as well as other establishments, and has demonstrated the flexibility envisaged by the modular design of the system. However, although it offers a straightforward and easy to understand approach to the management of filter processes, there are some underlying problems created by its simplistic approach. These are described in the following section.

6.3.3. Limitations of the Filter Chain Topology

Although the chain topology meets the requirements of the Microcosm model satisfactorily, and has provided a stable base on which to investigate various aspects of hypermedia functionality, there are a number of aspects of the system which are inefficient and that can cause problems. This section describes some of the problems that have been encountered.

6.3.3.1. Filter Ordering

The most apparent problem is the critical importance of the order in which filters are placed in the chain. If filters are not correctly located in relation to one another, then Microcosm may behave in an unexpected manner. For example the Available Links filter must be positioned after all Linkbase filters in order to collect all the links that are found and present them to the user. If a Linkbase were to be positioned after the Available Links filter then any links found in that Linkbase would be automatically activated without any choice on the part of the user. Similarly, the Link Creation filter must be placed in the filter chain before at least one Linkbase filter so that new links can be added to a link database.

This requirement can cause problems for users when they need to reconfigure the system, making the task harder than it need be. When moving an existing filter within the chain, or introducing a new filter to the system, the user must always take into account the function the filter is performing and how it relates to the other filters in the chain. In more complicated cases, it may even be necessary to be aware of the particular messages that a specific filter expects to receive, and those it will produce. Then, by comparing these details with the equivalent details for the other active filters, the filter can be placed correctly in the chain. For example the Show Links filter receives 'Show Links' messages and produces 'Follow Link messages.

Therefore, it must be located before any filters which would wish to receive these 'Follow Link' messages.

Further problems can be encountered if the intended functionality of a filter is particularly ambitious. For instance, a filter aiming to produce a map of the links emanating from the current document would ideally be able to query the active Linkbases to obtain the relevant information. This entails placing the filter before the active Linkbases in order for it to deliver its query. However, in order to receive the results of the query the filter also needs to be located after these Linkbase filters. In some cases, it is possible to implement a filter so that this is not necessary; for example, the current map filter learns about the links in the system as the user navigates, gradually becoming better at providing an accurate map. However, solutions such as this are clearly compromises which overcome the limitations of the current chain model.

6.3.3.2. Inefficient Message Passing

The second major problem encountered is the overhead in routing messages through the chain. A message making its way through the current filter chain does so by being sent to each filter in turn by the FMS, which has no notion of which, if any, of the filters in question are at all interested in that particular type of message.

This is illustrated particularly well by the basic configuration required to simply follow links. This configuration comprises a number of Linkbase filters followed by an Available Links filter with which activated links are presented to the user.

The link following process is initiated by the user selecting an item in a document and choosing 'Follow Link' from a menu. This results in a message being sent through the system to the FMS which in turn sends it to the first filter in the chain, one of the Linkbases. The Linkbase looks at the message and retrieves any relevant links from its database. These are sent back to the FMS along with the original 'Follow Link' message. All of these messages are then sent to the next Linkbase in the chain and so on. Eventually the 'Dispatch' messages describing the links found will reach the Available Links filter, and be displayed. This message passing is illustrated in figure 6.4.

The diagram shows how this process has resulted in the 'Dispatch' messages created by the Linkbases being sent to several other Linkbase filters which do not process them at all, but simply pass them back to the FMS, i.e. each 'Dispatch' message is sent to all those Linkbase filters which are located in the filter chain between the Linkbase in which the message originated, and the Available Links filter at the end of the chain.

Analysis of this excessive message passing shows that for *n* Linkbases, each returning *l* links from a query, the number of unnecessary message transfers, *t*, undergone by 'Dispatch messages before arriving at the Available Links filter is described by equation 6.1 below.

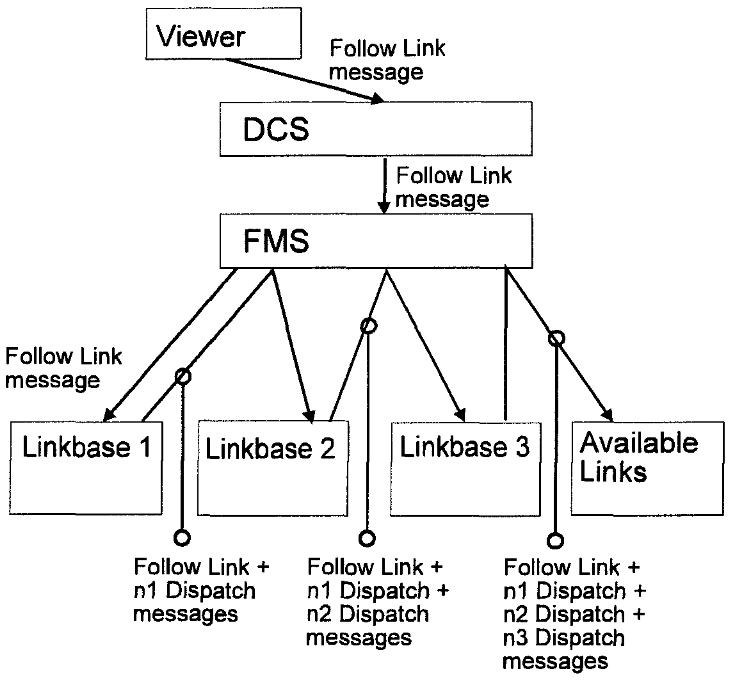

Viewer
Follow Link message
DCS
Follow Link message
FMS
Follow Link message
Linkbase 1
Linkbase 2
Linkbase 3
Available Links
Follow Link + n1 Dispatch messages
Follow Link + n1 Dispatch + n2 Dispatch messages
Follow Link + n1 Dispatch + n2 Dispatch + n3 Dispatch messages

Figure 6.4: The inefficient routing of 'Dispatch' messages through Linkbase filters which do not understand them (n_x is the number of links returned by Linkbase x)

$$\begin{aligned} &t = f(n) * l; \\ &where: \\ &f(n) = (n-1) + f(n-1); \\ &f(0) = 0 \end{aligned}$$

Equation 6.1: The number of unnecessary message transactions *t*, undertaken by 'Dispatch' messages in a configuration of *n* Linkbase filters, each returning *l* links.

This excess message transfer model is illustrated by the graph in figure 6.5. The function can clearly be seen to exhibit non-linear behaviour. The function's reference to the values obtained for previous arguments shows that it is based on the Fibonacci sequence.

Although this behaviour may vary a great deal in different filter configurations and with different message types, this graph shows clearly that, in a filter configuration with more than just a small set of filters, the performance of Microcosm can be expected to suffer. This is because the FMS is required to route each message through the same path, whatever the message type.

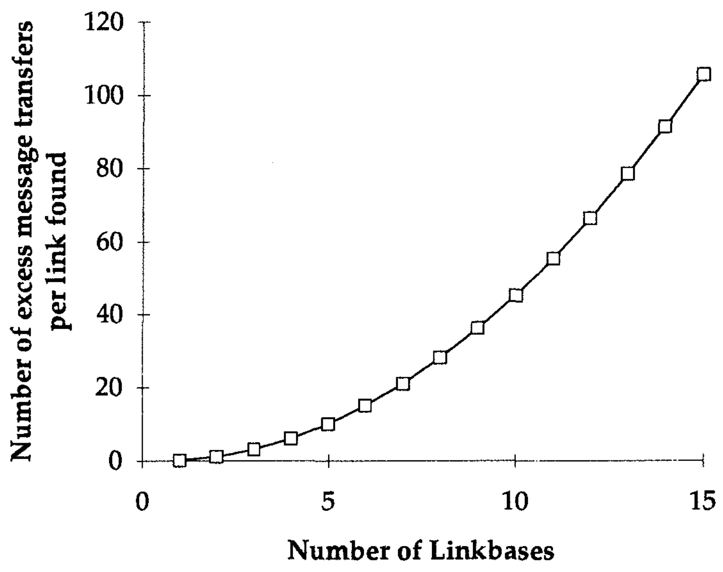


Figure 6.5: Theoretical graph of redundant transfers of 'Dispatch' messages in the simple, chain-based FMS as the number of link sources increases (each link source returns an identical number of links).

The system response described above is a worst case scenario, resulting from a complex configuration where all components are active. However, any system configuration beyond the simplest link following arrangement can be expected to suffer from this type of performance degradation to some degree.

In order to investigate the effect of these unnecessary message transfers further, experiments were devised to investigate the effect of various filter configurations on the performance of Microcosm. To do this, some components of the system were augmented to provide time stamps when they received messages: the FMS was altered to show the time it received a message from the DCS (i.e. the user issues an action request) and the Available Links filter was altered to show when it received 'Dispatch' messages. By establishing the time difference between the receipt of an action request by the FMS, and the receipt by the Available Links filter of the last link found, the response time of the system could be compared for different experimental configurations.

The response times presented in the following discussion were obtained as an average of several readings, with a minimum number of Windows applications running at the same time. This approach reduces the impact of any fluctuations in the underlying Windows environment which may, for example, yield control to other processes unexpectedly, or be called upon to load additional code into memory.

Although the range of possible configurations makes it difficult to create a definitive set of experiments, a range of scenarios were devised that show the effect on the system of certain, typical configurations. These different scenarios represent the basic aspects of a typical filter chain and can serve as a guide to the analysis of more complex system configurations.

The experiments described were all carried out using a single, IBM PC compatible workstation. This system was based on a 25 MHz Intel 486 processor, with 8MB RAM. This provides a suitable platform on which to run Windows 3.1 as consistently as possible. In particular, smaller RAM configurations can cause results to be distorted due to greater use of disk-based virtual memory. When performing the experiments, all non-essential Windows applications were terminated in order to prevent them from interfering with the execution of Microcosm. These measures prevent significant effects by external factors on the results obtained. It is impossible to completely prevent such effects since the execution of Microcosm is directly reliant upon the support provided by Windows. This reliance accounts for any slight deviations in the results recorded.

Case 1 : One Linkbase filter with Available Links filter

The first case is one of the simplest possible configurations. It consists of two filters, a Linkbase and an Available Links filter. The response time recorded for retrieval of a varying number of links from the Linkbase is shown in figure 6.6.

The experiments with the Linkbase filter carried out by Heath (Heath, 1992) show a response time directly proportional to the number of links retrieved, these experiments however were carried out with the Linkbase isolated from Microcosm. The graph below, achieved with the Linkbase installed within Microcosm also displays a linear response. This result can be predicted using equation 6.1 which shows that with one Linkbase, 'Dispatch' messages undergo no unnecessary transfers. This implies that for this simple configuration, the chain model provides the best path possible for messages; as the Linkbase generates 'Dispatch' messages, they are then immediately sent on to the Available Links filter.

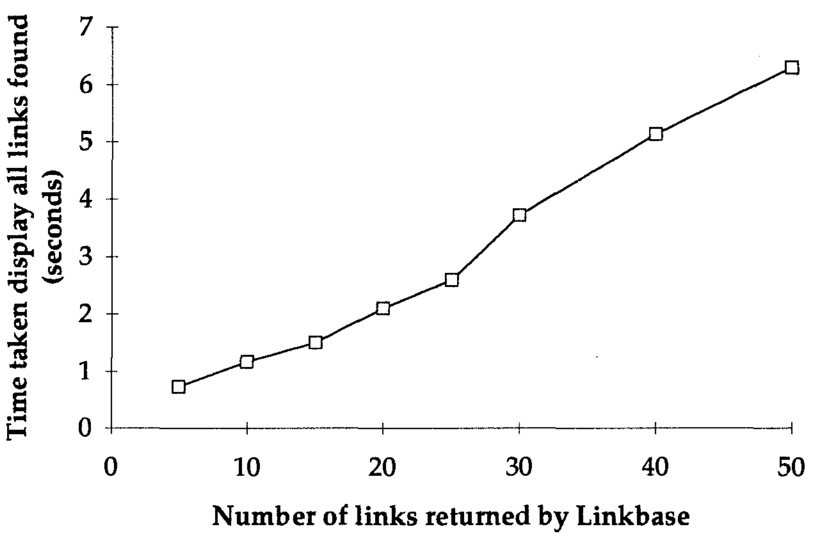


Figure 6.6: Graph of response time for the chain-based FMS with a simple filter configuration of one Linkbase and one Available Links filter

Case 2 : Multiple Redundant Filters

The second scenario investigated was that of a single Linkbase returning a variable number of links with a number of redundant filters present in the filter chain. A redundant filter understands messages other than those generated as a result of the processing of 'Follow Link' messages by the Linkbase, and is therefore required to simply pass these messages on to the rest of the chain, via the FMS.

Redundant filters were combined with the Linkbase and Available Links filters in two different styles of filter configuration. The first of these configurations located the Linkbase and the Available Links filters adjacent to one another in the filter chain, with a variable number of redundant filters added in front of them. The second configuration involved adding redundant filters between the Linkbase and the Available Links filters. The result of these experiments was to show how an interaction between specific filters could be adversely affected by the presence of other filters that have no actual role in the outcome of this interaction, and how the placement of such filters affects the performance of the system differently.

The results of the first configuration are represented by the graph in figure 6.7. Since the only message being sent through the redundant filters is a single 'Follow Link' message, the effect on response time is small, but shows a slight linear rise as the number of redundant filters increases.

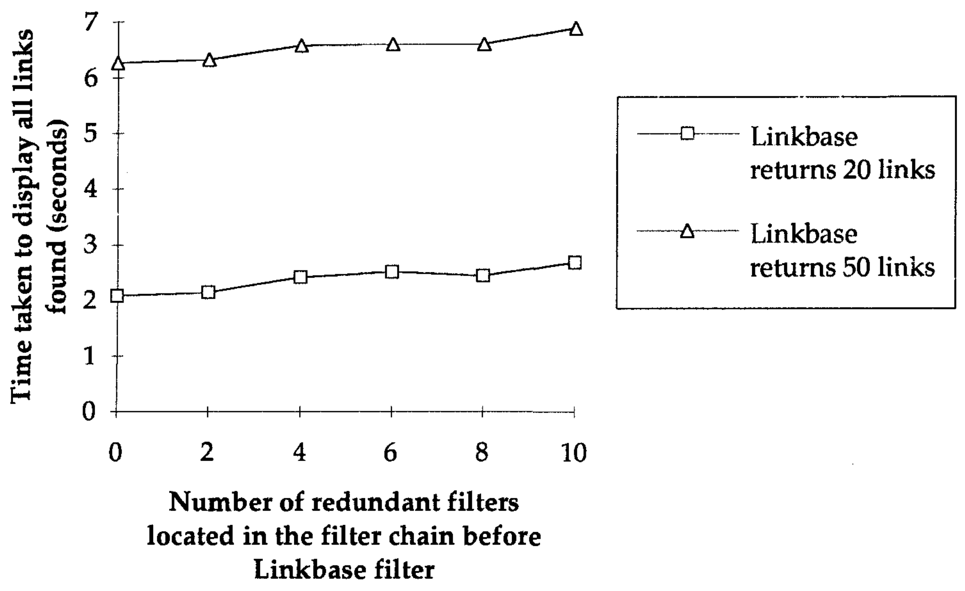


Figure 6.7: Graph of response time for the chain-based FMS when redundant filters are added before the Linkbase and Available Links filters

The results recorded for the second configuration reveal a more significant effect on the overall system response, as can be seen by the graph in figure 6.8. Initially, the graph shows a linear increase in response time. However, when more than four redundant filters are added to the filter chain, a threshold point is reached and a significant step in response time occurs.

Investigation of the problem shows that this is due to finite limits in the underlying Windows message delivery system, on which the Microcosm communications library is based. Each Windows application has a queue of finite size in which messages notifying the application of events may be stored. At the point at which this queue becomes full, and messages cannot be successfully sent, the Microcosm communications library is forced to build a queue of pending messages and attempts to deliver these messages again at timed intervals. As a result, the elements of the system which would normally be processing the current set of messages are left to become idle whilst the communication system gradually deals with the glut of messages. Further discussion of these details can be found in section 5.5.2 of (Heath, 1992).

As can be seen from the graph, when this queuing mechanism comes into effect, the performance of the system is immediately reduced.

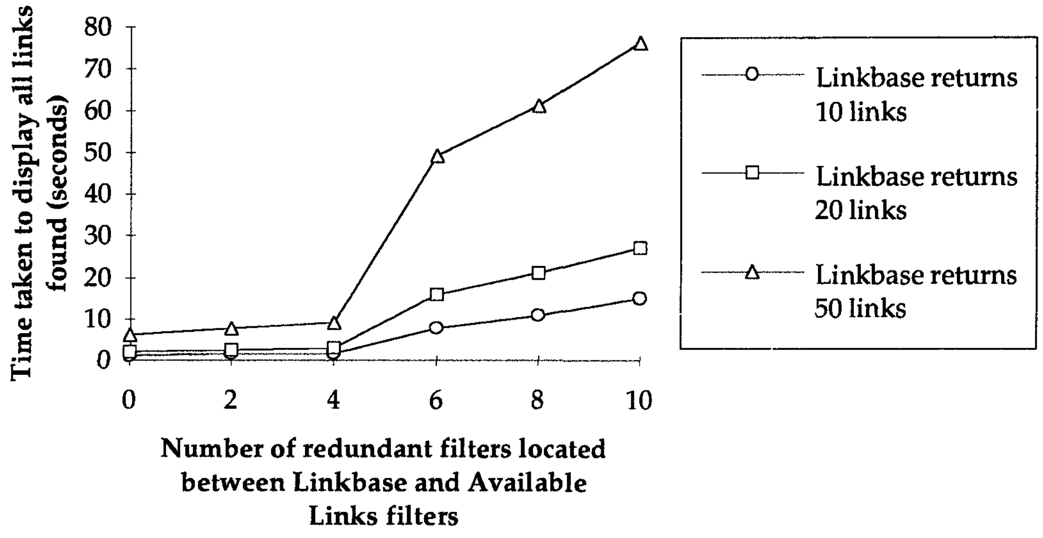


Figure 6.8 : Graph of response time for the chain-based FMS when redundant filters are added between the Linkbase filter and the Available Links filter

Case 3 : Multiple Linkbases.

This final case consists of several Linkbases, each of which returns the same number of links. The Available Links filter is located at the end of the filter chain immediately after all of the Linkbases. The resulting graph of system response time is shown in figure 6.9 below.

This graph can be seen to illustrate results similar to the non-linear increase in response time that was suggested by the theoretical analysis described earlier. This is to be expected, since the system configuration on which the analysis was based is identical to that used in this experiment. Of particular note, however, is the large increase that occurs in the 50 links per Linkbase series when more than four Linkbases are present. This step is again due to the Windows messaging system, and hence the Microcosm communications library, becoming overloaded with undeliverable messages and being forced to initiate its queuing process.

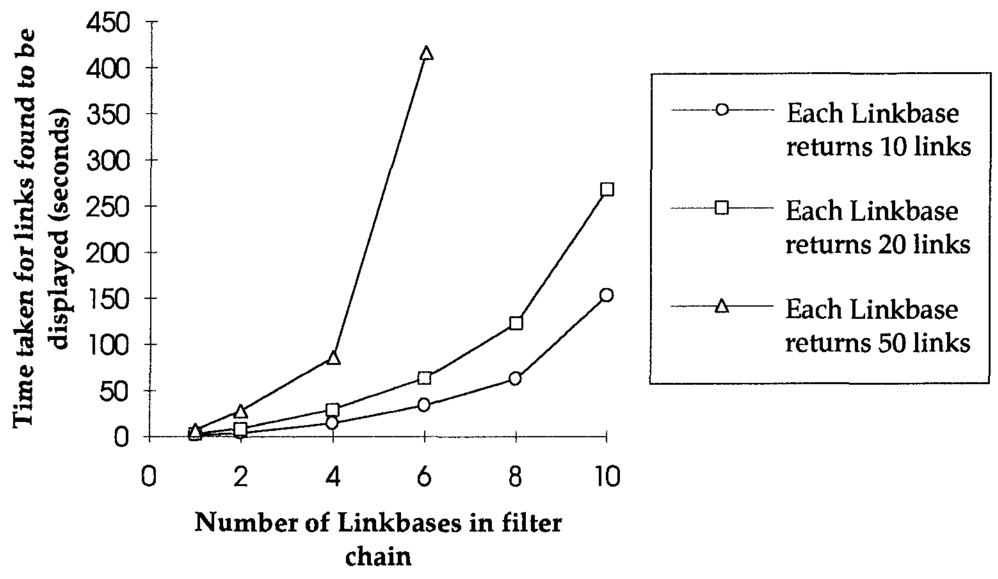


Figure 6.9 : Graph of response time for the chain-based FMS when multiple Linkbases all locate a certain number of links.

The results obtained from the experiments described in this section clearly illustrate that in many configurations, the simple filter model can restrict the overall performance of the system due to the chain approach to routing messages. This approach results in a message path which, for any given message type, can include filters which maintain no interest in that type of message at all.

6.3.3.3. Mapping the Microcosm model to a distributed system

It has always been intended that Microcosm will be extended in order to allow it to operate in a distributed environment, and therefore the processes of which the system is comprised must be able to accommodate any necessary extensions as simply as possible. In theory, the Microcosm model with its co-operative, inter-operating processes is fundamentally suited to such distribution. However, in a distributed system, the communication requirements of the separate processes will assume a more important role due to the additional overhead incurred by communicating between separate host systems.

Such overheads will clearly highlight any inefficiencies in the communication model, such as those described in the previous section. For example, if a number of Linkbase filters were located on various remote systems, the overhead due to 'Dispatch' messages being sent to these filters, only to be immediately returned unprocessed, would cause the problems illustrated in the experiments described to become even more pronounced.

6.4. Summary

This chapter described the advantages of the filter based model from which Microcosm derives its dynamic and extensible hypermedia functionality. By providing an open framework within which any number of different filters can be installed, the system can easily be extended to provide users with the tools they require.

The first implementations of this model were described, and experiments carried out to illustrate the performance achieved by the system. The limitations of this initial implementation were also discussed. The following chapter details the design of a more advanced FMS which addresses these issues, in particular the performance problem which would be encountered with a distributed system.

7. Advanced Filter Management

7.1. Introduction

The problems with the chain-based filter topology described in the previous section stems from the fact that this implementation is an essentially passive system. This means that on receipt of a message from the DCS, the FMS reacts in the only way it knows and sends the message to the first filter in the chain, and then subsequently to each of the other active filters in turn, without actually looking at the contents of the message. No attempt is made to guide the message efficiently to the appropriate filters, since the FMS is not provided with the appropriate knowledge of the functions carried out by the various filters installed in the system.

With a more efficient design, the FMS would be able to establish those filters to which a particular message must be sent, and route it directly to the correct filter or set of filters according to the type of action that the message represents. An enhanced FMS such as this would have two advantages:

- *More efficient overall message passing.* By sending messages of a particular type only to those filters that are able to process them, the overhead incurred by sending messages to filters that will not actually process that message type is removed. This improvement is particularly important for the extension of the back-end layer to operate in a distributed environment, where message passing becomes a more time consuming process and which must therefore be as efficient as possible.

- *Relaxation of ordering constraints.* By directing messages according to their type, the requirement for filters to be carefully ordered relative to one another is less important. There is no need to position filters producing certain message types before those filters that process these message types, since, wherever in the filter configuration a message of a particular type originates, it can be sent to those filters that are able to process it. Although the ordering problem still occurs in the case where more than one filter is present in the system that can process a certain message type, ordering is frequently less important in such cases. For instance, a typical configuration may have several Linkbases, all of which understand 'Follow Link' messages. As long as all the active Linkbases receive a 'Follow Link' message, the order in which they receive it is irrelevant. Most ordering problems occur because a filter which produces a particular type of message must be located correctly relatively to those filters which understand that message.

Such an approach can be considered to describe an 'active' FMS since it takes an active role in the routing of messages, unlike the original, passive, chain-based model.

The enhanced ability of such an active FMS to deal with filter processes on an individual basis is also important in the provision of a distributed system. It is unlikely that an entire filter configuration would be made available to a network; rather, certain interesting filters might be singled out to be utilised in this way. This facility would not be at all possible using the passive FMS, since it stored very few details about the available filters. The active system, however, is ideally suited to the provision of this functionality.

7.2. Active Filter Management Methods

As outlined above, an active FMS would be able provide an intelligent routing of Microcosm messages to filter processes. Such a system would offer an improvement in performance over the simple chain-based system already described by eliminating the redundant transfer of messages to filters which do not process that particular message type.

Figure 7.1 illustrates the problem that such redundant message transfers represent, by showing the logical path of messages generated by the passive FMS when a user attempts to follow a link (the physical path of the messages additionally includes the return of messages to the FMS, which then routes the message to the next filter). The message type that each filter actually processes is shown in brackets.

The diagram clearly shows how the 'Follow Link' message must first be passed through the Linker and Compute Links filters. The potential for increased inefficiency can also be seen further along the chain. The creation of a 'Dispatch' message, and its subsequent passage through any remaining filters in the chain once a link is found, shows that as new messages are created by filters, the number of redundant send/return cycles may increase rapidly. This redundant behaviour was illustrated by the graph in figure 6.5. The experiments described in section 6.3.3.2 clearly show that this behaviour can lead to a rapid breakdown in system response, even in straightforward filter configurations, due to the finite capacity of the Windows messaging system.

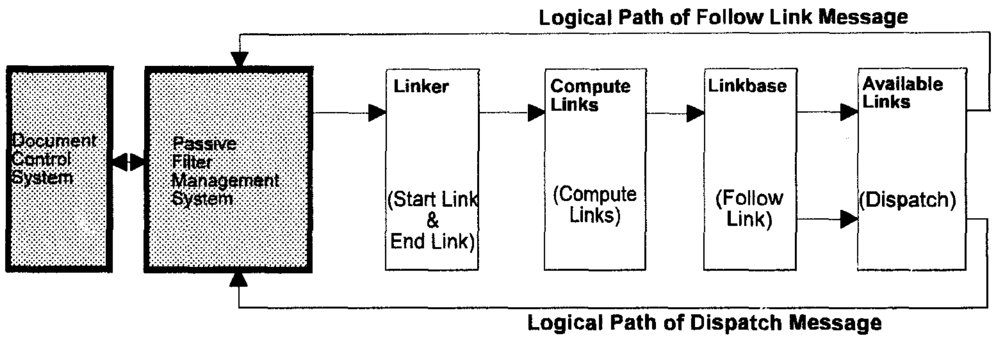


Figure 7.1: Logical message flow in the passive Filter Management System when a user requests a 'Follow Link' action. Note the redundant transfer of the 'Follow Link' message through Linker, Show Links and Available Links filters.

In the proposed active system, where messages can be routed according to their action type, filters must register the message types that they understand with the FMS when they are started. Using this information, the FMS is able to route messages only to those filters that understand them. This is illustrated in figure 7.2 which shows how the same 'Follow Link' message as in figure 7.1, and any messages subsequently created, would be routed using an active FMS. As before, the logical path of the messages is shown, and the physical path includes the return of messages to the FMS for routing.

The Compute Links and Linker filters are now no longer sent the 'Follow Link' message before the Linkbase filter can receive it. Also, the processed 'Follow Link' message returned by the Linkbase is no longer passed onto the Available Links filter. Because messages generated within the chain are also directly routed, the rapid increase in redundancy as the filter configuration becomes more complex is also removed. For example, another Compute Links filter positioned between the Linkbase and Available Links filters in the passive arrangement shown in figure 7.1 would receive the 'Dispatch' messages generated by the Linkbase, whilst these would be passed directly to the Available Links filter by the active arrangement in figure 7.2.

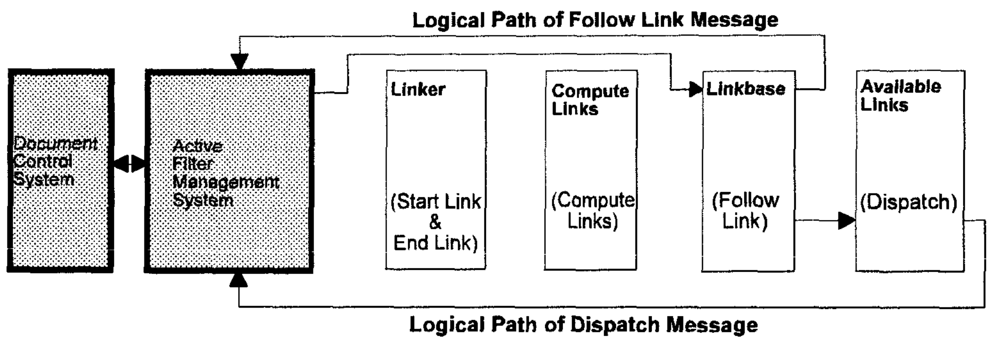


Figure 7.2: Logical message flow in the active Filter Management System when a 'Follow Link' message is received. Note that the 'Follow Link' message is routed directly to the Linkbase and to no other filters.

The results of the experiments described earlier, and the investigation into the possibility of an active FMS described above, both suggest that an active system would be effective in reducing redundancy in the back-end layer of Microcosm. Therefore a working version of the system was designed and implemented. The details of this development, and the analysis of the resulting performance is discussed below.

7.3. Initial Implementation

7.3.1. Filter/FMS Interface

The first step in the development of an Active FMS was to add the action registration phase to the process of starting a filter. In the simple chain system, the task of starting the appropriate set of filters is the responsibility of the FMS. When Microcosm is initially started, it executes the appropriate filters as specified in the various configuration files (see chapter five). If the user wishes to add filters to the default filter configuration at any point, they are able to use the filter configuration dialogue shown in figure 6.1.

In order to inform the active FMS of the actions they understand, filters simply need to be altered to send a message to the FMS once a communication channel has been established. However, whilst the nature of the FMS/Filter interface was being updated, it was decided to alter the nature of this filter execution process to improve the usability of the system.

In the current system, the FMS communicates with filters using the standard Microcosm communications library. Unfortunately the library is not very flexible in the way it allows connections between processes to be initiated. In order to connect two processes (for example, a filter and the FMS), each process must become aware of the pending connection and make a call to the communication library. When two processes have called the library, the communication library can complete the connection, and assign channel identifiers to the calling processes. In practice, this means that for a new Microcosm process to be started, it must be executed by the process to which it is to be connected.

This means that there is no way for the user to start a filter as they would other applications (e.g. through the File Manager or Program Manager), since when run in this way, the filter process would attempt to open a connection without the FMS having first initiated the connection process. An improved communication library would act in a more conventional client/server manner, with the central control processes (DCS, FMS) able to operate in a 'listening' mode so that they can be informed of connection attempts from potential child processes. Users would then simply start filters and viewers using their own preferred methods. Similarly, any applications that require a certain filter to be available would be able to start it simply by executing the process thus improving the flexibility of the system.

Although such functionality could also be introduced to the passive FMS implementation, the subsequent need to locate the new filter in the chain topology would negate the usability gained; the user would need to interact with the FMS anyway, in order to position the filter.

Unfortunately, this degree of change to the communications library would require all parts of Microcosm to be updated. In order to keep these changes within the scope of the FMS and filter processes, a simpler approach was devised which could be used by elements of the back-end layer in conjunction with the existing communication system. This demonstrates the benefits of removing the central control over process execution, whilst minimising the effort required to provide such functionality.

An additional Windows DLL was designed which could be used by both FMS and filter processes to provide more flexible connection. The interaction between this DLL and the connecting processes is illustrated in figure 7.3. When the FMS is started, it calls a function to initialise the library (point 1 in the diagram), giving a window identifier, and a Windows event message to be used to notify the FMS when connection events take place. When a filter starts, and wishes to connect to the FMS, it calls another function in the DLL (point 2), prior to attempting to open a communication channel as before (point 4A). As a result of this function call, the DLL notifies the FMS (point 3), using the pre-specified Windows event, that a new connection is pending. The FMS can then make the appropriate call to the standard communication library to complete the connection (point 4B). At this point, the calls to the communication library can complete, and the FMS and filter are each returned a communication channel identifier (points 5A and 5B).

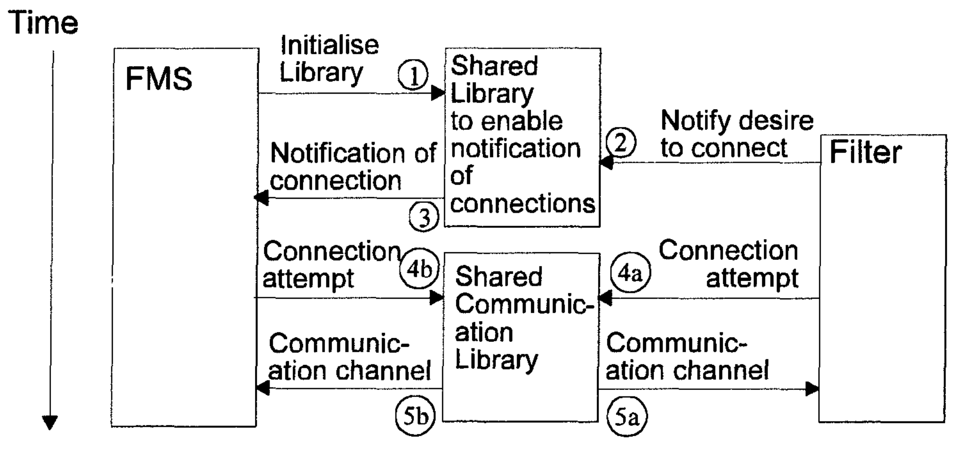


Figure 7.3: The new filter connection process used by the active FMS.

When it calls the DLL, a filter also supplies a message detailing the types of message that it wishes to receive which is passed on to the FMS. This is used to build the infrastructures which enable active message routing to take place.

7.3.2. Message Routing

The messages supplied by filters when they are started provide all the details needed by the active FMS in order for it to be able to route messages, present configuration information to the user, and to allow filter configurations to be stored for re-use. These details include a list of actions that the filter understands, a caption to identify the filter to the user, and the command line string necessary to execute the filter again should the user wish to store a filter configuration for reuse.

This information is used by the active FMS to build two data structures with details of the filters that are currently installed. The first is a simple array containing a list of the filters that are active, with all the basic details about each filter, such as the actions it supports and the filter's caption. The initial definition of this filter structure is shown in the code fragment below.

```
typedef struct {
   char szCmdLine[MAXCMDLINE]; /* command line that runs filter */
   char szCaption[MAXCAPTION]; /* caption identifying filter    */
   char szActions[MAXACTIONSTRING];/* list of actions supported */
   CHAN chan;                  /* id of communication channel   */
   } FILTERDETAILS ;
```

The *szCmdLine* field specifies the command line required to run the filter again, and allows the FMS to recreate a stored filter configuration. The *szCaption* field is used to identify the filter to the user, and the action list specifies the actions understood by the filter, each action being separated by a comma. The *chan* field is the identifier of the Microcosm communication channel connecting the FMS and the specified filter.

The second data structure is the action table, this stores the list of filters that are registered for each message type. The table is used by the FMS to route messages to the appropriate filters. It is a 2-D data structure which is based on an array of FILTERCHAIN structures that identify each available action. Each of these structures contains a further array of ACTIONDETAILS structures, describing the action-specific attributes of the filters which process that particular action. The ACTIONDETAILS arrays are analogous to the filter chain in the passive FMS, but in the active FMS, there are many of them, each valid for only one message type. The definitions for these structures are shown below.

```
typedef struct {
   UINT uFilterIndex ; /* pointer to filter list          */
   BOOL bOn ;          /* is filter active for this action? */
   } ACTIONDETAILS ;

typedef struct {
   char szAction[MAXSTRINGLEN+1];/* which action is this       */
   UINT uFilters ;              /* how many filters in array   */
   ACTIONDETAILS FAR *lpFilters ;/* ptr to action's filter array*/
   } FILTERCHAIN ;
```

The FILTERDETAILS structure can be seen to be very simple: recording just the action in question, the number of filters currently processing that action, and a pointer to the list of these filters. The ACTIONDETAILS structure is just as simple, it allows a filter to be toggled on and off for the action in question, and any further information is accessed through the *uFilterIndex* field which points to the appropriate entry in the FILTERDETAILS array described earlier.

As the filter configuration changes, the ACTIONDETAILS arrays are dynamically resized to minimise the amount of memory used. The FILTERDETAILS array however cannot be contracted when filters are shut down, since this would render references to it in the action table invalid. Instead, the *chan* field is set to -1 to indicate that the record is unused. This space can then be re-used if a new filter is subsequently added to the system. Figure 7.4 illustrates the relationship between the various data structures described above

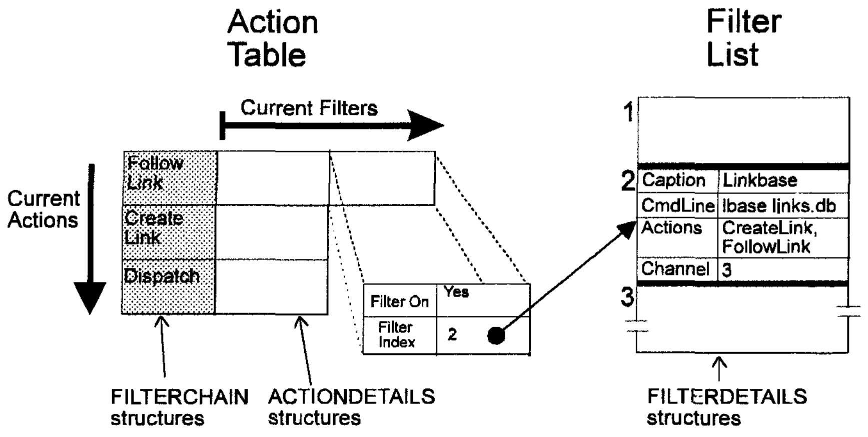


Figure 7.4 : Conceptual model of relationship between the data structures used by the Active FMS to store filter details

When the active FMS receives a message of a particular type, the action table is searched to locate an entry for that action. If such an entry is found, the message is then sent to the first element of the action's filter chain. Within this chain the message is treated exactly as in the passive FMS and gradually progresses through the chain of appropriate filters. However, the message will not be sent to any filters that do not process the required message type. When a message reaches the end of a chain, it is echoed back to the DCS, as are messages of a type for which no filters are available.

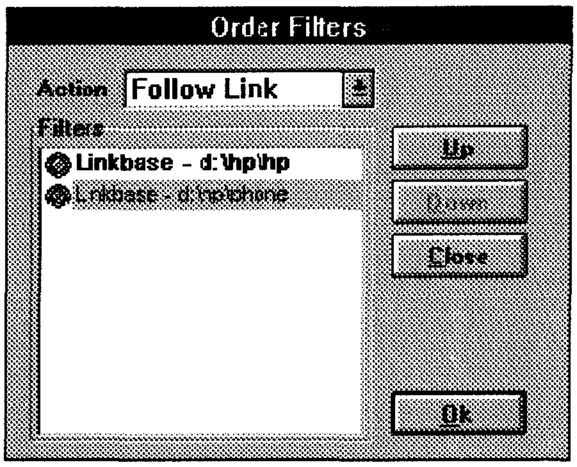
If a filter produces a message of a type which it does not process, then there is no defined 'next filter' for the message to be sent to in any of the action sub-chains. Instead, the message is sent to the first filter in the chain for its particular action. For example, when processing 'Follow Link' messages, a Linkbase may generate one or more 'Dispatch' messages. These would be redirected to the chain of filters which understand such messages, for example the Available Links filter.

The filter chains for each type of action are built dynamically as filters are executed. New filters are appended to the end of the appropriate filter chains by default, but may be altered by the user if necessary (see below). A particular configuration can be saved by the user for future use, in which case all the filter positions and options that are described by the filter table data structures are maintained.

7.3.3. Other Aspects

The information now available to the FMS allows much greater flexibility to be incorporated into the user interface of the system. Whereas before the only configuration facility was the option to alter a filter's position in the filter chain, each filter in the system may now be configured separately for each message type it supports. This includes the ability to toggle any individual action supported by a filter on and off. An example of this function in action is the ability to temporarily stop a particular Linkbase searching for links, by turning off its support for the 'Follow Link' action. The Linkbase can be just as quickly returned to normal. This allows much finer tuning of the functionality provided by a particular set of filters than was possible in the passive system.

This configuration functionality is provided by the dialogue box shown in figure 7.5. This allows the user to select which of the current available actions is to be configured, by way of a drop-down list. The diagram shows the 'Follow Link' action selected. When an action is selected, the various filters available for that action are shown in the large list box. The example shows two Linkbase filters available. Each of these filters may be reordered within the list using the 'Up' and 'Down' buttons, or terminated using the 'Close' button. This offers the choice of terminating just this action (i.e. remove the filter for this action only), or terminating all supported actions (i.e. close down the filter completely). The small 'lights' to the left of the filter descriptions in the list are used to indicate whether the filter is currently 'on' or 'off' for this message type. This is shown by the colour of the light, green for on, red for off, the caption of a disabled filter is also 'greyed out' to reinforce the notion of inactivity. Filters can be toggled on and off by clicking on these lights in the dialogue.



Action: Follow Link

Filters:
- Linkbase - d:\hp\hp
- Linkbase - d:\hp\phone

Figure 7.5: Filter configuration dialogue for the active Filter Management System.

This centralisation of configuration information also allows the general interface to Microcosm to be improved. In the passive filter management model, no part of the system was aware of the actions offered by the available filters; even the FMS only knew how many filters were active, and their descriptions. As a result, viewers were forced to list all possible Microcosm actions that they were aware of in their menus, in order for the user to be able to invoke these functions. Should a particular action not be available from the current filter configuration, the user could still make a selection in a document, choose the action from the menu, but nothing would happen. Conversely, viewers had to have new menu items compiled into them in order to support any filters that provided a new action.

As the FMS now stores information about the available actions, a system whereby viewers can dynamically create menus with only those actions that are currently available is possible. This was done by using an existing Microcosm library that had been developed for this purpose. This library was originally designed to operate with the passive FMS, and allowed individual filters to register those actions that they wish to appear in a menu with this central service. Viewers could then obtain an accurate menu. This library is used by the active FMS to provide viewers with appropriate menus. By requiring the FMS to deal with registering actions, the menus obtained can reflect any configuration of the current filters, and reduces the tasks which filters must carry out.

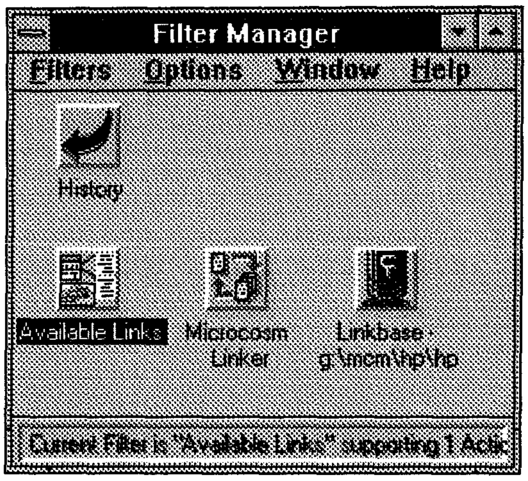
In order to allow the FMS to keep track of the additional information required to register and de-register menu actions correctly, the FILTERDETAILS structure updated to store additional information. The *szActions* field with its list of supported actions was replaced by an array of MENUACTION structures, each entry describing one action supported by the filter. This structure identifies the action in question, determines whether it should be available in menus, and if so, the description to be used and the document types it is applicable to. The MENUACTION structure and the updated FILTERDETAILS structure are shown below.

```
typedef struct {
   char szAction [31] ;                  /* action string        */
   char szDescription [MAXSTRINGLEN] ; /* menu string          */
   char szTypes[MAXSTRINGLEN] ;        /* types supported      */
   BOOL bMenu ;                          /* is it a menu action */
   } MENUACTION ;

typedef struct {
   char szCmdLine[MAXCMDLINE];   /* command line to run filter   */
   char szCaption[MAXCAPTION] ; /* caption identifying filter   */
   CHAN chan ;                    /* id of communication channel */
   UINT uActions ;                /* no. of actions supported    */
   MENUACTION FAR *lpActions ;   /* array of action descriptors */
   } FILTERDETAILS ;
```

Another advantage of the new FMS implementation is the interface it presents to the user. With the previous system, filters were allowed to behave as normal applications in the Windows environment. Each filter had a visible window, although this would usually remain minimised. This leads to a problem, particularly in systems which have a low resolution display, where the number of filter processes active lead to the icon area at the bottom of the screen becoming very cluttered.

This problem has been addressed by the active FMS, which searches for each filter and obtains the icon used by the process. This is then used to provide a centralised filter display as a window of the FMS, and individual filter processes may hide themselves. This has many advantages: all filter processes are logically grouped together, the users' display becomes less cluttered, and the FMS interface may take on a more directed form, rather than relying on descriptions which the user must relate to the icons on the display.



Current Filter is "Available Links" supporting 1 Actic

Figure 7.6 : The improved user interface of the active Filter Management System. The icons of the installed filters are removed from the desktop and presented in a separate window. This allows them to be hidden when not required by minimising the FMS window.

This improved display is shown in figure 7.6, user interaction with the icons is passed onto the filter processes as if the user were interacting with the filter process itself. The only disadvantage of this interface is the lack of control the filter process has over its icon display. For example, in the passive system the Linkbase filter was able to animate its icon to show that it was searching.

7.4. Analysis of Performance

Once the implementation of the active FMS was completed, the experiments described in section 6.3.3.2 were repeated in order to allow the performance of the active and passive Filter Management Systems to be compared.

Case 1 : Linkbase and Available Links

Once again, for this simple configuration, the response time of the system was found to be essentially linear. As with the equivalent configuration for the passive FMS, all the installed filters take an active part in the transaction being analysed. The graph of the results for this configuration is shown in figure 7.7. The results for the same passive system configured with these filters are also reproduced for comparison.

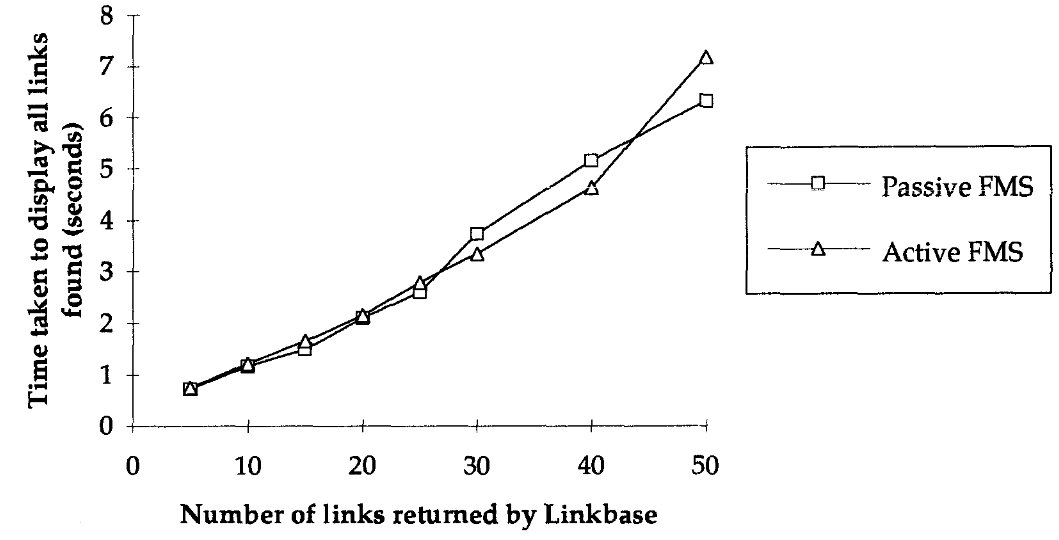


Figure 7.7: Comparison of the system response time of the active and passive Filter Management Systems. The results shown are for a simple configuration of one Linkbase and an Available Links filter.

The graph shows that for this simple case, the performance of the two systems is very close. It is difficult to draw concrete conclusions from the small differences measured between the two systems, since fluctuations in the underlying Windows environment cannot be satisfactorily eliminated; however the performance of the active system can be seen to be approximately equivalent to the passive system. This can be accounted for by the extremely simple configuration being evaluated.

In this configuration, the active FMS employs its message routing algorithm to deliver the messages it receives to the appropriate filters, but in this particular case the linear chain arrangement is able to provide an identical path for the messages generated, with only one extra message transfer (in the passive system, the 'Follow Link' message returned by the Linkbase is sent to the Available Links filter, whilst in the active system it is not). The active system, however must check the type of each message, and route it accordingly rather than simply sending it blindly to the next filter. Thus both systems have a small overhead in routing the message to the appropriate filter.

Case 2 : Multiple Redundant Filters

This configuration is more representative of a 'real-life' filter configuration, since additional filters are present that do not have an active role in the link following transactions used as a basis for these experiments. The experiments carried out with the passive FMS (section 6.3.3.2.) required two potential filter configurations to be investigated to show the effect of variant positioning in the filter chain. The results obtained show a response time that varies from a slight linearity to an overload of the Microcosm communication system, depending on the particular configuration and the demands being made of it.

With the active FMS it is not possible to contrive alternative configurations in this way. In the context of the FMS, a redundant filter is one that does not process the messages that it receives as part of a particular interaction. In the passive FMS these filters share a single filter chain with those filters that do play a part, but the active FMS is designed to remove such redundancy from the system. Thus only one configuration was required to illustrate the performance of the active FMS with redundant filters present in the configuration.

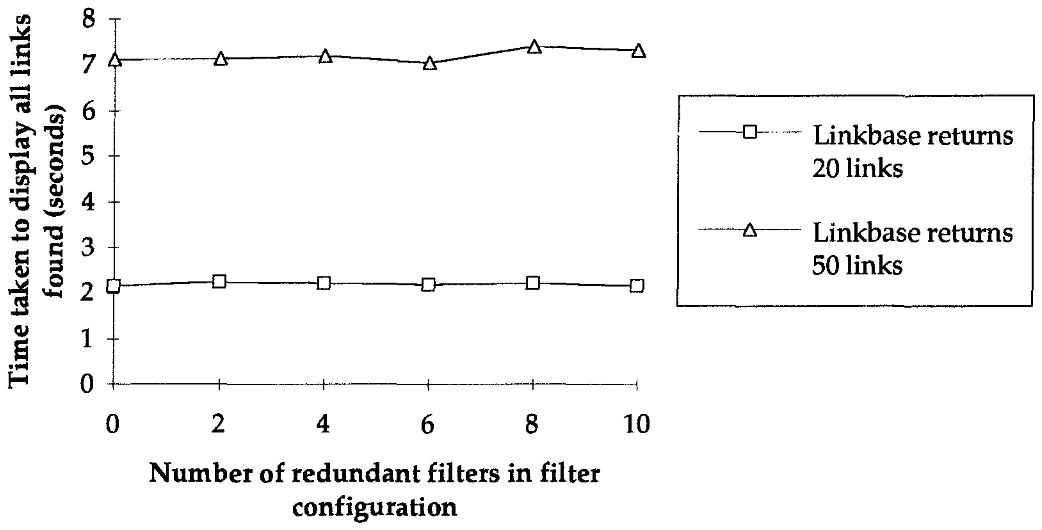


Figure 7.8: Constant system response time for active Filter Management System with redundant filters in the configuration.

The graph in figure 7.8 shows the behaviour of the active FMS for a filter configuration containing redundant filters. Since the active system only routes messages through the appropriate filters, the system response remains constant as the number of redundant filters is increased. The only factor effecting response time is the number of links being retrieved. This represents a small improvement over the results for the passive system shown in figure 6.7 which showed a slight linear response as redundant filters are added at the beginning of the filter chain. It is however a significant improvement over the results in figure 6.8, which show a large performance penalty for the case where redundant filters are added to the passive system between interacting filters.

The results obtained for this configuration show a significant performance improvement for the active system, clearly bettering the non-linear behaviour of the passive system which is caused by the excessive, redundant message passing that the active system eliminates.

Case 3: Multiple Linkbases

The final scenario to be evaluated was that of a configuration containing multiple Linkbases, each returning links. Again the results that were achieved by the active FMS show improvement over the passive system. The graph in figure 7.9 shows the performance that was recorded.

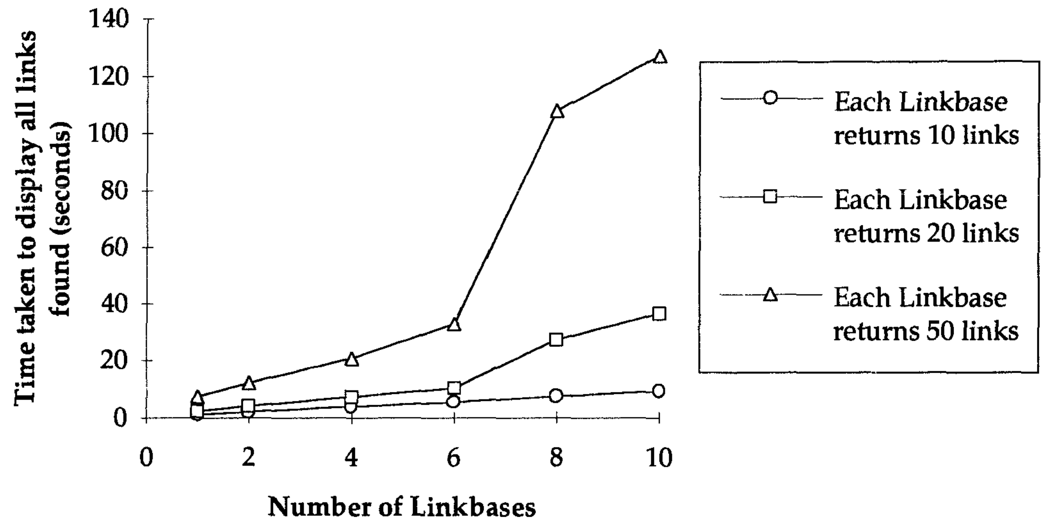


Figure 7.9 : Response time of Active FMS for a configuration containing multiple Linkbases

The passive FMS produced a straightforward, non-linear response for this type of configuration (see figure 6.9). The performance of the active system however is clearly not as simple. Looking at the results for up to 6 Linkbases, the response achieved can be seen to be linear. This is the response that one would expect since the 'Dispatch' messages received from each Linkbase are routed directly to the Available Links filter, and do not undergo redundant routing through the other Linkbases present.

This linear performance however does not last beyond the 6 Linkbase configuration. After this point, a significant step in response time occurs which can be seen in the 20 and 50 link graphs. This dramatic degradation in performance is due to the breakdown of the underlying Microcosm communication library, which is overwhelmed by the sheer number of 'Dispatch' messages that it is required to deliver. This causes its back-up message queuing system to be activated.

Since the performance of the system does not degrade until approximately 120 links are being located (6 Linkbases, each returning 20 links), this result can be considered a boundary condition which is unlikely to effect a typical user. If a user is in a position where this many links are being generated, then a greater problem can be expected as he or she attempts to determine the appropriate links to follow. Also, since the basic response is linear, the performance degradation can be reduced by paying specific attention to the bandwidth of the underlying communication system. This would not be possible for the passive system, where the non-linearity is a characteristic of the basic message traffic generated.

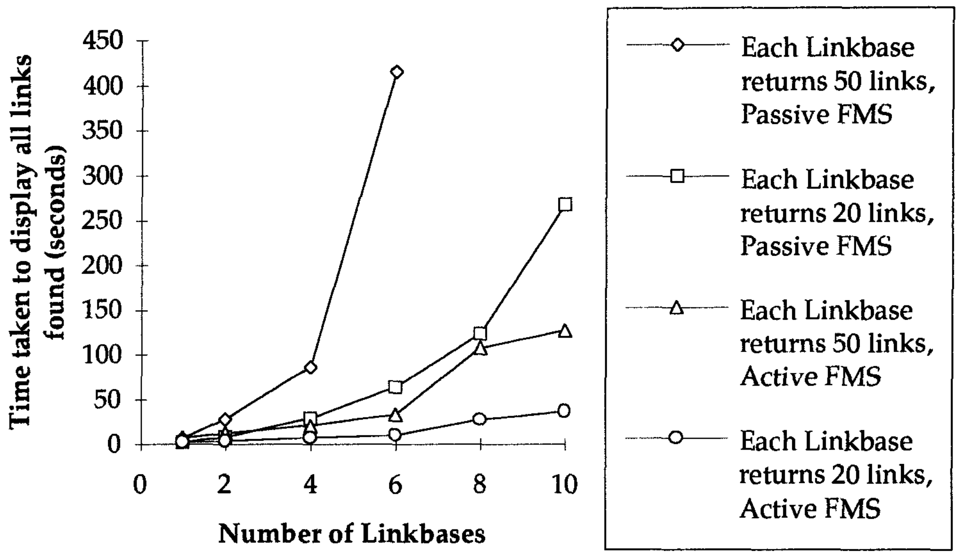


Figure 7.10: Comparison of the equivalent response times for the passive and active Filter Management Systems with a configuration containing multiple Linkbases

The performance offered by the active FMS is still a significant improvement over the equivalent configuration of the passive FMS, despite the problems that have been described. This can be seen in figure 7.10 above which shows the 20 and 50 link response times for both active and passive routing methods to allow direct comparison.

The results of these experiments show clearly the improvement in performance that has been achieved through the adoption of an active filter management approach. In fact, for the active FMS, the results obtained are closer to those that would be found in a typical 'real-world' configuration. This is due to the way that it treats filters in an independent and discrete manner, only receiving those messages which they understand. This is not possible with the passive system, which treats the entire filter set as a single entity. In a particular real world configuration, it might be possible to identify instances of each experimental configuration within the single chain. To analyse the performance in such a case, the passive system must be treated as a compound system, consisting of several interdependent sets of interacting processes.

As well as the performance improvements that have been shown, the more informed and structured approach taken by the active FMS makes it easier for each filter to be treated individually. This allows the interface to the FMS and the rest of Microcosm to be improved as described in section 7.4.2.3. It also provides a basis for the implementation of a distributed system using filter processes as the basic elements of distribution, since the problem of communication overhead described in section 6.3.3.3 has been significantly reduced by the active FMS.

7.5. Summary

This chapter has described the development of the 'active' FMS, which addresses the limitations of the initial passive FMS, described in chapter six. The improved performance, flexibility and interface of the new system was also discussed. The following chapter describes how the filter handling capabilities of this active Filter Management System can be extended further, in order to create a distributed version of Microcosm.

8. A Flexible Architecture for Distributed Hypermedia Systems.

8.1. Introduction

It is apparent from the discussion in chapter four that the distributed hypermedia systems that have been developed in the past have not been true open systems, and the distributed models used by these systems have not been sufficiently flexible to allow a wide range of configurations to be achieved. This chapter discusses the design and implementation of extensions that have been made to the Microcosm system, described in chapters five, six and seven, to allow a truly open system, with flexible support for a distributed environment, to be created.

Chapters six and seven describe in detail how the modular filter architecture which provides Microcosm's hypermedia functionality allows the system to be easily, and dynamically, configured to support particular user requirements. The aim of the distributed version of Microcosm is to provide this configurability in a networked environment, and to allow a complementary level of flexibility in the distributed model created.

The flexibility of the distributed hypermedia systems described in chapter four was restricted due to the fixed distribution of system components permitted by the distributed models they use. For example, the HAM system, KMS and Hyperbase require a central server to provide all information to users. This results in a very strict implementation of the client/server model. The other systems described also display this strong distinction between the client software utilised by users, and the server software that provides the central storage and management of information and links.

The aim of the distributed version of Microcosm described in this chapter is to provide a system which is capable of operating in a wide range of ways, rather than in just one restrictive arrangement. Although a simple client/server configuration would be possible, it is just one of many possible system configurations.

8.2. Requirements for an Open, Distributed Hypermedia System

For a hypermedia system to be categorised as an open system according to the definition given in chapter two, it must allow functionality to be distributed on a network:

A system in which data and system processes can be distributed across a network. A system operating on one host should be able to share hypermedia information with any other systems running on other hosts. Although the hosts attached to the network may represent heterogeneous hardware platforms, open communication protocols will allow them to interoperate transparently.

This definition allows a range of interpretations of the meaning of sharing hypermedia information, but obviously, for heterogeneous platforms to be integrated, an open protocol based upon a widely available network communication system is a basic requirement.

Another aspect of these requirements which is relevant when considering the functionality of a distributed hypermedia system is the ability to incorporate existing tools:

In particular this modularity should be able to provide integration of existing tools, rather than requiring users to change their approach to suit the functionality that the hypermedia system provides by default.

Although this requirement is primarily concerned with integrating local tools, it also applies to the incorporation of other distributed systems. For example, a user who wishes to utilise the facilities of distributed information services such as WAIS and Gopher should be able to use their particular hypermedia system in conjunction with these facilities, to add links and other additional information to the stored information.

The flexible, open model provided by Microcosm is well suited to meeting both of these requirements:

- Existing information services may easily be incorporated as either filters or viewers depending on the type of functionality provided. For example, the browsing-oriented interface of Gopher is best suited for use as a viewer. As such, it might offer 'meta-viewer' functionality, launching other Microcosm viewers to display the information located. A query-based system such as WAIS, on the other hand, is more suited to incorporation as a filter; this would allow functionality similar to that of the Compute Links filter to be provided, but using large distributed information resources rather than a set of local documents.

- The distribution of Microcosm's own functionality obviously requires extensions to the existing system; however, the modular nature of the system architecture is an ideal basis for the provision of such extensions. Hypermedia functionality can be distributed at the filter level, allowing discrete components of the total functionality to be made available, whilst access to documents can easily be incorporated, since all access is made through the Docuverse system. This system can trap those calls which refer to remote systems, and take appropriate action.

The discussion of distribution in this chapter is concerned primarily with the extensions to the Microcosm architecture described above, for further discussion of the integration of existing services into the Microcosm frame work see (Hollom, 1993).

8.3. Design of the Microcosm Distributed Model

8.3.1. Basic Elements of the Distributed Version of Microcosm

The essence of the Microcosm model is its provision of hypermedia functionality as a set of discrete components (filters). This separation of the various aspects of the available functionality provides an obvious starting point for the distribution of hypermedia information. The flexibility of configuration provided by the filter model can be immediately inherited by the distributed version of the system; for example, a particular subset of the current filter configuration could easily be made available to remote systems, and similarly, remote information could easily be assimilated into the local configuration.

In order to provide such functionality, the Microcosm model needs only to be extended to allow local filters to be 'published' so that other systems may utilise them, and to allow remote published filters to be connected to the local system. The local FMS must then utilise a well defined protocol to transfer messages to remote systems when necessary. Similarly, it must be prepared to receive messages from remote systems, either in reply to a message sent to a filter within that system, or a message to be delivered to a published filter in the local filter configuration. Obviously, to achieve this addressing of messages to individual filters, the active FMS described in chapter seven is required, since it has access to information about the various filters available at any particular time. This allows the required extensions for a distributed system to be added to the FMS, as a central resource.

The system described above allows links, and other forms of hypermedia connectivity, to be accessed from a remote system. The result of such activity is likely to lead to references to documents available on the remote system. Therefore a complementary extension to the system is required to allow such documents to be retrieved. Since the Docuverse system described in chapter five provides all document access for Microcosm, this is a logical place for such an extension to be made. In order to determine whether a document identifier refers to a local or remote file, the identifier must be extended to incorporate details of the host system. Then, access to remote files can easily be detected and carried out using a standard protocol.

8.3.2. Flexible Distribution Using Filters

As described above, the basic element of distribution for Microcosm is the filter. In the simplest case, two users may collaborate through the use of shared Linkbases each providing the other with links to annotations and personal documents. However, since filter processes represent a very small system granularity, a full range of other distributed models is possible through the use of this basic system. These are summarised in table 8.1.

The table shows how, by varying the approach taken to publishing and connecting distributed filters, a whole spectrum of models is possible. Systems based upon distinct client and server components, such as HAM and KMS, can easily be simulated by configuring particular Microcosm systems as either server-only or client-only. Similarly, it is still possible for Microcosm to act as a stand-alone system as described in chapters five and six. There is no system-enforced requirement for the distributed facilities to be used, they simply offer an enhancement to the basic functionality of Microcosm.

Distributed Model for local system	**How the system operates**
No Distribution	The system operates as described in earlier chapters, with no interaction with remote systems
Client Only	The system operates purely as a client, with all hypermedia functionality provided by a remote server(s)
Server Only	Alternatively the system might make all of its functionality available to remote systems and disable local interaction with the system, operating purely as a server.
Peer to peer interaction	Finally, the system can combine both client and server functionality within the same configuration. This creates a network comprised of equal, peerless systems.

Table 8.1: The range of distributed architectures available by utilising the flexibility of the Microcosm distributed model.

In between these two extreme examples of the functionality available lie a whole spectrum of more interesting possibilities. These are all based upon the peerless nature of the model that is achieved, where any particular Microcosm system on a network may act as an information server. This feature immediately resolves the problem faced by many of the systems described in the previous chapter, which made no allowance for such a requirement. Thus, a client system can access many servers at any one time, each of which may provide a different form of hypermedia connectivity, or which may offer links relating to a different subject area.

Beyond this scenario is the situation where each Microcosm system on a particular network is capable of acting as both a server and a client system within the same configuration. This brings much more flexibility than the strict client/server transactions provided by contemporary systems, offering scope for many forms of collaboration between users, or workgroups. Additionally, since all these configurations are possible within the same system, and each is based on a well-defined, open protocol, all possible configuration variations may co-exist. Thus any particular system can be configured in such a way to suit its particular purpose, whilst not excluding its use by other system configurations. The diagram in figure 8.1 illustrates the wide range of configurations that have been described, and shows how the various possible configurations are able to interact.

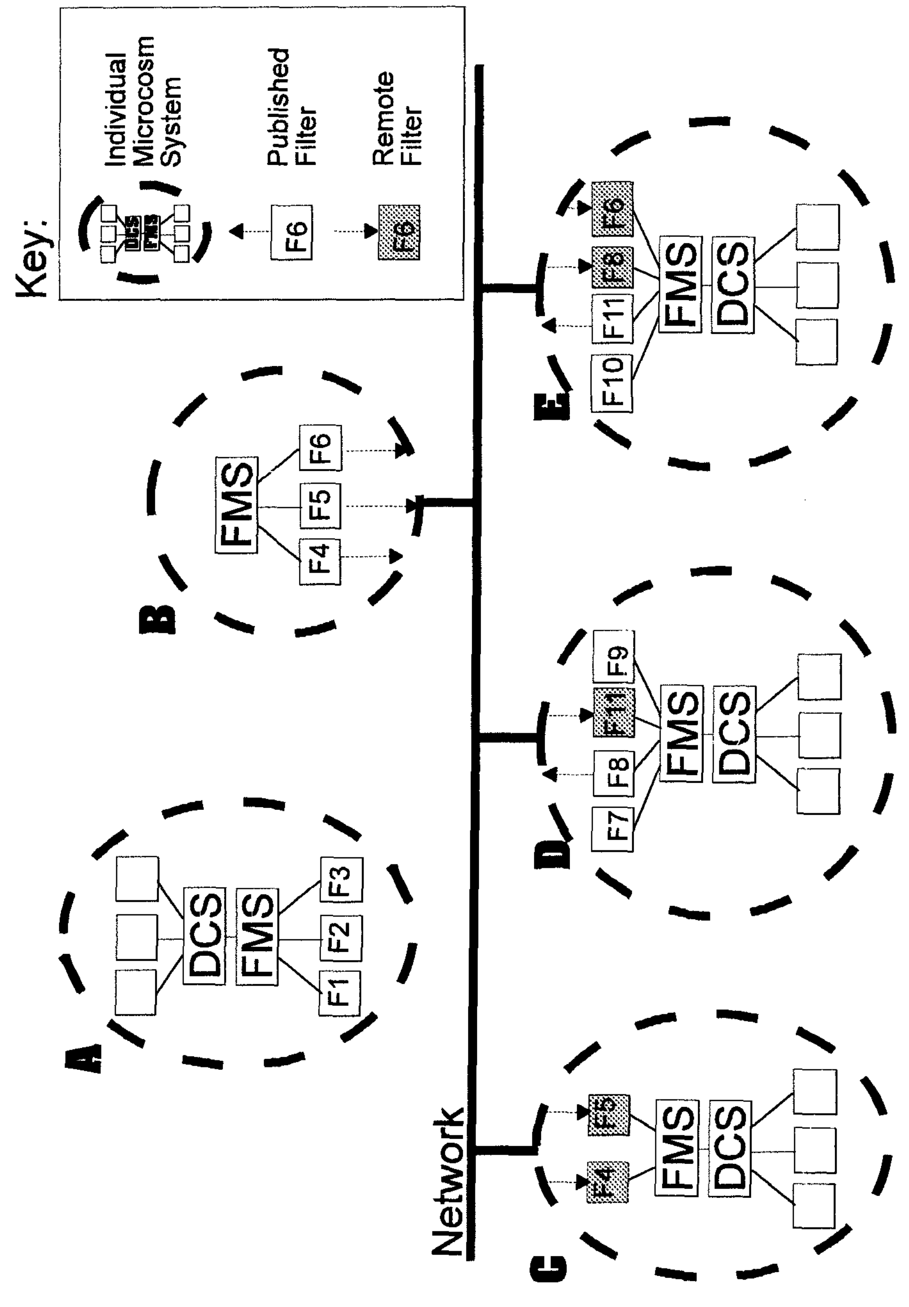


Figure 8.1: The range of possible filter configurations available using the Microcosm distributed model.

System A shows Microcosm operating in a stand-alone manner, as described in chapter five. System B shows Microcosm acting as a hypermedia server with no local interaction. System C shows Microcosm acting purely as a client, obtaining all hypermedia functionality from remote filters (in this case provided by system B). Finally, systems D and E show two Microcosm systems operating in a peer to peer manner, they are sharing filters to allow cooperation between the systems in addition to incorporating local filters and remote filters from other Microcosm systems.

8.3.3. Extending the Microcosm Docuverse

The main extension required of the Docuverse system, in order to support the extensions to Microcosm described in the previous section, is the ability to distinguish document identifiers which refer to remote systems, from those which specify documents that are available on the local system. The current document identifier uses the date and time that a document is imported into the Docuverse to generate a unique identifier for the document within the local system. In order to identify the network location of a document, it is necessary to identify the unique network identifier of the host machine in addition to the local document identifier.

Since the Universal Resource Locator (URL) scheme used by the World Wide Web is becoming widespread as a means of describing remote documents and the access method they require, it was decided to utilise this syntax for Microcosm document identifiers. Because of the free indexing used by the underlying database system, changing the identifier format requires only that the database index is rebuilt, and therefore converting existing Docuverse databases to this format is straightforward.

The document identifier is now formed as follows:

<access method>:\\<network name>[:<port number>]\<locally unique id>

For Microcosm documents, the access method is set to "mcm", but by incorporating this field other access protocols may also be supported by Microcosm (see section 8.4). Typically, the local unique id for a Microcosm document will be a Docuverse identifier, but it can also be a simple file name; this is the key difference from the standard WWW URL definition. The advantage of this approach over the standard

URL form is that the document may be moved or renamed on the local system but still be accessible from remote systems, since the Docuverse on the host system will record any change of location. With a standard URL, hard-coded document references can easily become out of date if files are moved, a factor which forces the WWW to remain a rather static system.

To utilise this extended document identifier, the Docuverse function that creates a new identifier must be updated, and any functions which use such an identifier must extract the access method and host details to check whether the document being referenced is present on the local system. If not, a standard protocol is used between Docuverses on distinct systems to transfer the appropriate file, or its Docuverse record.

8.3.4. Providing an Open, Heterogeneous System

One of the key requirements of the distributed version of Microcosm described above is its ability to operate in a network environment comprised of heterogeneous hardware and software platforms, a process that is often fraught with incompatibilities and inconsistencies. In order to do this, the communication and connectivity techniques used must be as widely available as possible. For this reason the development has been based on standard network protocols. However, the actual design is not reliant on such protocols being available, and the network specific layers of the system could easily be altered to support other platforms. The only requirement is for some form of general purpose, connection-oriented network transport layer to be available.

The implementation that is described in this thesis is based on some of the most common network and communication standards, and notably the same standards used by the Internet network and the many information services which it supports. The physical network used locally is Ethernet (Comer, 1991) although the development described is at a much higher level and would operate transparently over other physical networks simply by modifying the underlying hardware and software services. The network protocol used is TCP/IP (Comer, 1991), which is the basic transport layer used by the whole of the Internet network.

The only aspect of the underlying network to directly affect the implementation of the distributed version of Microcosm is the actual communication protocol used between instances of the system. The Socket protocol is used for its widespread availability on many platforms (the protocol is a basic element of the UNIX operating system, and is available for both IBM compatible and Macintosh personal computers) and the fact that it is already used by a large number of existing Internet information services (e.g. WAIS, Gopher, WWW, Archie). Although other protocols may offer better solutions for particular platforms (NetBIOS for IBM, Appletalk for Macintosh), no other protocol is available for such a wide range of systems.

A particular factor in this decision was the timely conception of the Winsock specification (Hall et al., 1993) for the provision of a socket programming interface for the Microsoft Windows environment within which the majority of Microcosm development takes place. This led to the availability of a large number of reliable socket programming libraries for use with proven PC network solutions (e.g. LAN Manager, PC-NFS, Novell NetWare).

A common problem when carrying out network communication between heterogeneous systems is that of the byte ordering of binary data, however, since the Microcosm message format (see chapter five) is text based, this is not a problem in this particular instance.

The choices described enable the system to remain as open as possible in terms of interoperability between different implementations of Microcosm, and the straightforward integration of existing network services.

8.4. Implementation Details

This section describes how the Microcosm distributed model was implemented, and details how the FMS and Docuverse components were extended to provide the required functionality.

8.4.1. Extending the Active FMS

8.4.1.1. Data Structures for Distributed Filter Management

The first step in the implementation of the model described in the previous section was the extension of the data structures containing information about the set of active filters. There are two new requirements for these data structures, to accommodate the distributed model:

- specify the availability of a particular local filter to remote systems,
- incorporate a reference to a remote filter within the local filter configuration.

The first of these requires that additional information be incorporated within the data structures; the resulting changes to the FILTERDETAILS structure (which describes the general configuration of a filter), and the ACTIONDETAILS structure (which describes action-specific configuration of a filter), are shown below. The original definitions of these structures are described in sections 7.2. and 7.2.3.

```
typedef struct {
   char szCmdLine[MAXCMDLINE];  /* command line to run filter   */
   char szCaption[MAXCAPTION] ; /* caption identifying filter   */
   CHAN chan ;                  /* id of communication channel  */
   UINT uActions ;              /* no. of actions supported     */
   UINT uNetworked ;            /* general networking supported */
   MENUACTION FAR *lpActions ;  /* array of action descriptors  */
   } FILTERDETAILS ;

typedef struct {
   UINT uFilterIndex ; /* pointer to filter list            */
   BOOL bOn ;          /* is filter active for this action? */
   UINT uNetworked ;   /* form of networking supported      */
   } ACTIONDETAILS ;
```

The additional *uNetworked* fields allow a hierarchical specification of the permitted distribution of the actions supported by a particular filters. The field may be set to one of three values which are summarised in table 8.2 below:

Value of Field	**Meaning**
NETWORK_NONE	Indicates that the filter is local to the system and not networked in any way (or private)
NETWORK_PUBLISH	Indicates that the filter is local to the system, but is 'published', allowing it to be utilised by remote systems
NETWORK_REMOTE	Indicates that the filter does not reside in the local system. The entry in the filter table structures represents details of a virtual connection to a remote system

Table 8.2 : The possible values of the *uNetworked* field in the filter table data structures

At the general level, described by the FILTERDETAILS structure, the overall distribution of the filter is defined. At the action level, the *uNetworked* field in the ACTIONDETAILS structure allows this general setting to be overridden for that particular action. For example, a Linkbase may distribute all of its actions except the 'Create Link' action, preventing remote users of the filter from adding further links to the database. To create this configuration, the *uNetworked* field for the FILTERDETAILS structure describing this Linkbase, and the *uNetworked* fields for each supported action except 'Create Link', would be set to NETWORK_PUBLISH. The 'Create Links' action entry would have this field set to NETWORK_NONE, in order to indicate that it should not be made available to remote systems. Altering these networking settings is accomplished through a number of functions which set the fields of the various affected structures appropriately.

The requirement to incorporate a remote filter into the filter table may be met within the existing data structures. In order to reference a remote filter, two properties are required: the network name of the host system, and a reference for the particular filter required. The latter property is provided by the filter caption, which already exists as a field (*szCaption*) of the FILTERDETAILS structure. By using the caption rather than a numeric identifier, the system is protected from changes in the remote system. For instance, if the remote system is restarted, the description will be constant whilst a dynamic identifier might change. The remote hostname is stored in the command line field (*szCmdLine*) since the execution information normally stored there does not apply in the case of a remote filter. As described in table 8.2, the *uNetworked* field is set to NETWORK_REMOTE to indicate that the filter is not resident in the local system.

8.4.1.2. Handling Inter-System Message Transfer

With these augmented data structures in place, it is now possible to create distributed filter configurations such as those illustrated in figure 8.1. In order for them to operate however, a mechanism by which messages may be transferred between discrete systems is required.

This mechanism was created using the recently created Windows Sockets API (also known as *winsock)*, a programming interface agreed upon by the major suppliers of TCP/IP networking products for the Windows environment. It is designed to allow applications to be developed in such a way that they are immediately portable across a variety of networking platforms. The developments described in this thesis were implemented using the PC-NFS system from Sun Microsystems, but were also tested for portability using other available Winsock-compatible systems. Socket programming is described at a very general level in this chapter; a detailed description of this topic is provided by (Renaud, 1993). The way in which the general socket programming approach has been modified to suit the Windows environment is described in (Hall et al., 1993).

The message transfer facility was implemented using Internet Stream sockets, the reliable and sequenced form of the socket protocol. In order to use these sockets, a process on one system must create a socket, and connect it to a *server* socket on a remote system. This is done by specifying the network address of the remote system, and the *port number* of the server socket. The server socket may then accept the connection request and complete the connection by reciprocally creating a socket in the server process. Once the connection is made, bi-directional communication between the two processes may take place.

In the Microcosm distributed model that has been described, each Microcosm system is able to act as either a client or server process, in a peer to peer manner. This entails that each system must act as a server in order to receive messages from other hosts. Equally, each server system must be able to act as a client in order to reply to messages they receive. The basic unit of communication in Microcosm is the message, and this is the level of granularity at which inter-system transfer must take place.

To facilitate this message transfer, each Microcosm system creates a server socket which 'listens' for connections at a pre-defined port. This allows each system to receive messages from other systems. To send a message, a system simply connects to the remote system, and sends its message. The remote system acknowledges the receipt of the message with a simple handshake reply, and closes the connection.

Once a message has been delivered, the system which receives it must direct it to the appropriate filter to be acted upon. In a purely local system, this routing activity is handled by the message type and the knowledge of which local system component the message was received from. If a message is received from a remote system, it may be intended for any one of the filters which are published within the local system. Therefore, some additional information is required to identify the target filter of the message. This is provided by adding an additional tag

('FMSFilter') to messages that are sent to remote filters. This tag contains the description of the filter provided when the remote connection to the published filter was created (see next section). This allows the filter to which the message relates to be identified, and the appropriate action to be taken. If the identified filter is a local filter which has been published, then the message is destined for the filter and is sent to it for processing. However, if the filter is defined to have the network property NETWORK_REMOTE, then the message is in fact a reply from the remote system, to a message that has previously been sent to a remote filter. In effect the message has been received from the 'virtual filter' entry in the local filter table, which corresponds to the actual filter on the remote system. In this case, the message has its additional tags removed, and is routed in exactly the same manner as if the filter it was received from was in fact local.

An additional requirement is for a system to return a remote message to its source once it has been processed. In order to do this, a further tag ('MCMServer') is added to the message to identify the system from which it originated. When the FMS receives a message from a filter with this tag, it is identified as a remote message which has been processed and must be returned to the source system. If the filter generates additional messages as a result of processing a remote message, it adds these new tags to the messages, in order to cause them to be sent to the remote system.

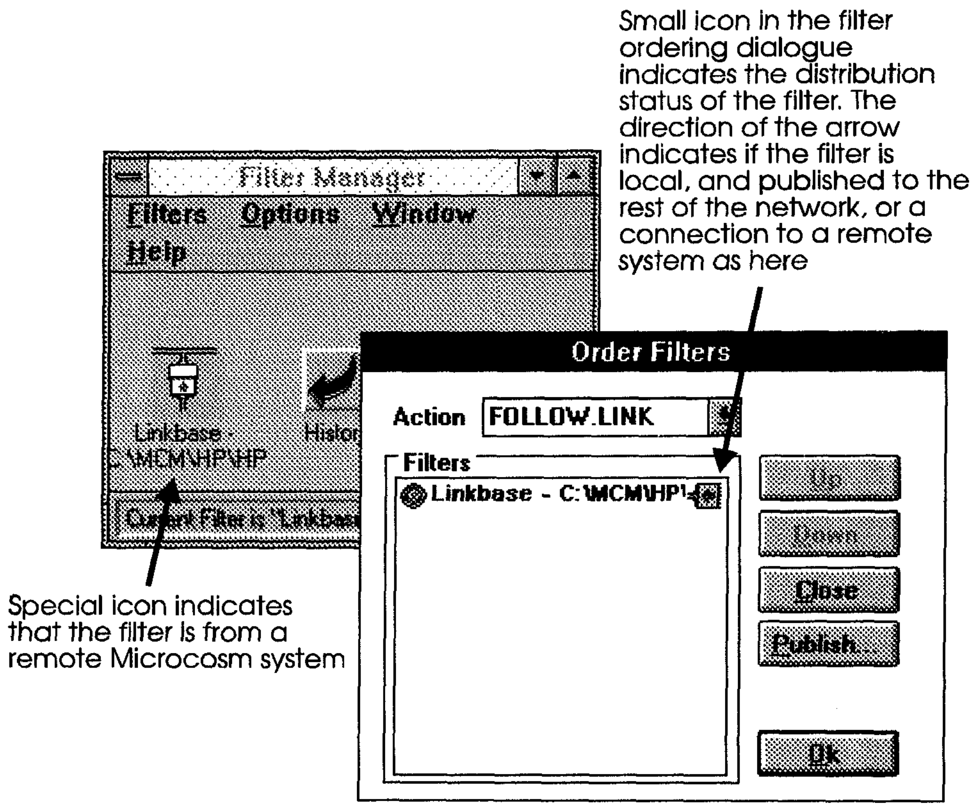
To support this message transfer process, a queuing system was required to address the imbalance between the speed at which messages are delivered for sending, and the speed at which they may be transferred. This queue prevents the message transfer code from being flooded with transfer requests. Instead, messages to be sent are simply placed in the queue, which is continually checked by the message transfer code under control of a timer event. If a message is found in the queue, and a message transfer is not already underway, it is removed from the queue and sent to the appropriate remote system, where it can be routed to the appropriate filter.

The final piece of additional information incorporated into a message serves as a precaution against the situation where the local filter configuration contains remote filters with duplicate descriptions, for example where two remote servers offer similarly named filters. It is possible for this to cause a permanent loop of message transfers between systems, particularly if they process the same actions. If a message is sent to the first of these filters, and then eventually returned, it will then pass along the chain until it meets the duplicate filter. When the message is returned from the duplicate, the filter table will be searched for the specified description from the 'FMSFilter' tag. This search will find the first instance, and send the message to the next filter, starting an infinite cycle of messages around the duplicate filters. Because the search is based solely on the caption, the system is unable to distinguish between messages received from the duplicate filters.

In order to address this problem, a final extra tag ('FMSFilterIndex') was defined that identified the filter table index of the source filter in addition to the filter's description. This allows a particular Microcosm system to correctly identify the remote filter from which a message has been received, no matter how many filters may share the same description. When the message is sent to the remote system, this index is incorrect since it specifies the filter's location in the local configuration, and in this case it is overridden by the use of the caption. When messages are returned to the local system however, the message can be directed immediately to the appropriate section of the filter configuration. This system overcomes the problem of infinite cycles in a system connecting to remote filters with identical captions. A related problem would exist in a server with duplicate *published* filters, whereby messages are always delivered to the first instance, however, this situation is unlikely to arise since a server is not expected to provide more than one published filter with the same description.

The final requirement is for additional measures to deal with any errors which may occur. If a message cannot be sent, either because the remote system to which it was targeted could not be contacted, or if the destination filter is no longer available, the filter reference in the local table is bypassed and the message is sent to the next appropriate filter.

These inter-system message routing facilities complete the basic framework of the distributed version of the active FMS. Figure 8.2 shows the additions made to the FMS interface to communicate details of the network status of filters. Remote filters are represented by a special icon in the main window, whilst in the filter configuration dialogue the small icon in the list box indicates that a filter is networked, with the direction of the arrow dependent on whether the particular item is a published local filter, or a virtual connection to a remote filter.



Small icon in the filter ordering dialogue indicates the distribution status of the filter. The direction of the arrow indicates if the filter is local, and published to the rest of the network, or a connection to a remote system as here

Special icon indicates that the filter is from a remote Microcosm system

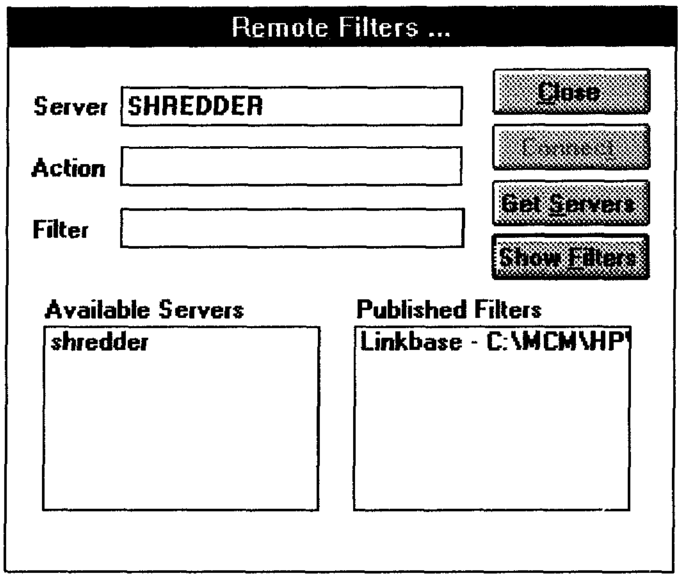
Figure 8.2: Filter configuration dialogue illustrating the notation used to communicate network status of filters.

8.4.1.3. Locating and Connecting Remote Services

The previous section describes how the distributed version of the active FMS can support cooperation between networked Microcosm systems. However, the initial difficulty for the user is how to locate remote Microcosm systems which may have published filters to offer. To provide this facility, an additional inter-system communication protocol is defined to allow the following information to be determined:

- Does a particular host offer a Microcosm server?
- Does a particular server have any published filters?
- What are the details of a particular filter, so that a virtual connection to it may be created?

Unlike the inter-system message transfer which takes place between established Microcosm servers, the reliable stream socket is not appropriate. Instead, the User Datagram Protocol (or UDP) socket type is used. This allows distinct packets of data to be delivered, but is an unreliable service which does not require a connection to be obtained prior to sending information. This makes it ideally suited for delivering queries to systems in order to establish whether they provide particular services. To support this form of querying, the FMS provides a second permanent socket, which is bound to another well known port number to accept these UDP queries. Queries are formed as Microcosm messages, which are then sent to the appropriate system(s). Replies to queries may be provided if necessary, or possible, via the same UDP facility. In the current implementation of the FMS, the user utilises the dialogue box shown in figure 8.3 to interact with these facilities.



Server: SHREDDER
Action:
Filter:

Available Servers: shredder

Published Filters: Linkbase - C:\MCM\HP

Figure 8.3: The dialogue used by the distributed FMS to allow the user to access filters available via remote systems.

If the user has no idea of the servers which are available, or that they wish to use, he or she may select the 'Get Servers' button. This causes a number of 'Are you a Microcosm server?' queries to be sent to remote hosts, depending on the configuration of the FMS. The basic approach is for a query to be broadcast on the local network, all local systems will receive this message, and those with a Microcosm server running can understand it and reply appropriately. The systems that are located are listed in the 'Available Servers' list of the dialogue box. As well as broadcasting messages, the system looks in the current configuration files for lists of known servers which will also be contacted. These additional hosts may be located world-wide as well as locally. Thus, the 'Available Servers' list will contain all known servers which are active, as well as all those in the current network.

Once servers are identified, the user is able to select a server, and request details of any filters that they publish simply by choosing the 'Show Filters' button. This causes a 'Show Filters' message to be sent to the server in question and a reply which enumerates those filters available is returned. These are displayed in the 'Published Filters' list. To make a connection to a filter, the user selects it in the list box and chooses the 'Connect' button. This causes a 'Connect Filter' message to be sent to the appropriate server specifying the filter to which a connection is being requested. The reply to this message contains all the details necessary for the filter to be incorporated into the local filter table, as described in sections 7.2.2 and 8.4.1.1.

As an alternative to the step-by-step query interface style described above, the user may simply type the details of a known server directly, in order to access the 'Show Filters' function, or enter both server and filter information to access the 'Connect' button directly. The FMS can also use these functions on its own behalf, for example, if a stored filter configuration is being loaded, the system will attempt to carry out a connection request in order to check that the remote filter is still available, and can provide the same set of actions as when its details were saved.

8.4.1.4. User Feedback

When using Microcosm on a single system, the user is able to receive visual feedback from the various filters that are present, and hence be aware of what the system is currently doing. This feedback can take many forms, such as animation of icons or changing the mouse cursor. Unfortunately, this functionality is lost when connections to remote filters are made, since the remote filters communicate only through Microcosm messages.

To afford feedback to the user that some action is actually taking place, some visual clues have been incorporated in the networking code described in the previous sections. Whenever reading information from, or writing it to, the network, the cursor is changed to one which includes a network symbol. This shows the user of a system with remote connections that his system is still processing messages, and also allows a user of a server system to be aware that published filters are being accessed.

8.4.2. Extending the Microcosm Docuverse

8.4.2.1. Changes to Docuverse Functions

Section 8.3.3 describes the extension of the Docuverse identifier to allow the specification of remote documents. In order to provide the appropriate support for these extended identifiers, the basic Docuverse access functions must be extended to enable them to access remote systems. These are listed in table 8.3 below.

Function	Purpose of Function
QD_IndexGot	Retrieve a document's details from the Docuverse using its Microcosm document identifier
QD_FileGot	Retrieve a document's details from the Docuverse using its filename
QD_OpenFile	Open a document, given either a filename or Microcosm document identifier

Table 8.3: Docuverse functions which were extended to provide access to remote systems.

Each of these functions was extended in some way to provide appropriate functionality for the distributed system. For the QD_IndexGot and QD_FileGot functions an attempt to access the local Docuverse is made initially in case the document specified is local, or it is a remote document with information stored locally for some reason. If this fails, a URL parsing function is used to break the document identifier into its component pieces, and the host server specified is compared with the local hostname. If this differs, the remote system is accessed and the appropriate details retrieved. If the host is the same, then the function returns an indication of failure since the specified document was not found by the earlier local query.

The QD_OpenFile function operates in a slightly different manner, it immediately breaks the URL down to establish the location of the specified document. If the document is local, the appropriate details are retrieved from the Docuverse, and the standard Windows OpenFile function is used to provide a file pointer. Otherwise, the remote system must be accessed and the file data spooled to a temporary local file. This can then be opened as normal, and a file pointer returned to the calling process. Standard Docuverse file I/O functions are then used to operate on the local copy of the file.

8.4.2.2. Incorporating Extensible Access to Diverse Information Systems

The modified URL system can obviously allow many different network access methods to be specified within the system. For example, a link to a document stored by a Gopher server will specify an access type which identifies the Gopher file transfer protocol. Although these documents are not stored by a Microcosm Docuverse, the function of the Docuverse is to provide document access for the whole of Microcosm, and is therefore the most appropriate place for access to alternative information systems to be provided.

In order to make this aspect of the Docuverse extensible, the various access protocols are implemented using separate Windows DLLs. The appropriate DLL for each access type is specified in a table in the system configuration file, and this allows the correct library to be dynamically loaded as required. These libraries must offer a support function (with a pre-defined name) for each of the functions in table 8.3, which may be accessed as necessary by the Docuverse. The parameters to the function specify the various elements of the URL descriptor (access mode, host, port number, path).

The libraries are treated as 'black boxes', and the exact interpretation of the required functions is left open, since it may vary according to the particular protocol being provided. The DLL's role is to act as a mediator between the requirements of Microcosm, and the underlying models of other systems, as encapsulated in their access protocols. For example, the WAIS system is based on the Z39.50 protocol (Lynch, 1991), whilst the World-Wide Web is based upon the use of the HTTP protocol (Berners-Lee, 1993a). The modularity gained by this approach allows for the growth of the system to encompass future information systems which may appear, and to utilise changing network standards.

At present, the only DLL available implements the Microcosm specific 'mcm' access protocol. Like the network extensions to the FMS, it uses the support provided by the Windows Sockets standard as a basis for its development. TCP/IP stream sockets are used to support the protocol described in table 8.4.

The Docuverse supports these queries through the use of a TCP server socket on a well known port. When remote systems connect via this socket, the Docuverse receives requests specifying the appropriate details, and returns the result to the calling system on the same socket. If the request is for a document, then the document size is returned first in a fixed size block. This allows the receiving system to indicate the progress of the transfer as it receives information.

Type of request	QD_FileGot, QD_IndexGot	QD_OpenFile
Step 1	Connect to remote docuverse via well-defined port number	
Step 2	Send Message describing particular request, and document to perform it on	
Step 3	Receive document details from remote server.	Receive document size, followed by actual document data

Table 8.4: The protocol used by the Docuverse to access documents stored on remote Microcosm systems.

This protocol is very close to that used by many other TCP/IP based information systems (e.g. Gopher, World-Wide Web), and could easily be extended to accommodate these protocols as well.

8.4.3. Improving Performance

In order to improve the performance of the system, documents retrieved from remote systems are cached locally. If the user attempts to return to a remote document that was previously retrieved, they are first offered the choice of viewing the cached version. If the document is likely to have changed, they may choose to retrieve a more recent version. At the end of a session, the cache is purged, and local copies of documents are removed from the system. This approach can greatly improve the response of the system, in particular when retrieving multi-media information such as images and sound.

Because the vast majority of information that is accessed via a network is in a static state, problems with cached documents are likely to be rare and of low impact to a users' work. However, if the system is being used to support collaborative work, this optimistic approach is easily compromised unless the collaborators are in close contact. Possible approaches to solving this problem are described in chapter nine.

In addition to the caching of documents, some investigation was carried out into improving the performance of the distributed FMS. As described earlier, communication between FMSs located across a network is carried out at the level of a Microcosm message; however, it is often the case that several messages must be sent to the same remote FMS. For example, on receipt of a 'Follow Link' message from a remote Microcosm system, a Linkbase may create several messages which each describe a possible link, each of which must be returned to the originator of the 'Follow Link' message.

In an attempt to optimise this aspect of the system's performance, a system for caching socket connections between systems was devised. Rather than terminate a connection once a message had been delivered, the details of the connection (the remote system and socket number) were stored in a fixed size cache. As the cache filled, old connections would be closed and new ones added. To allow for a variety of potential configurations the size of the cache was user-configurable, for example to allow a server-oriented system to maintain a greater number of connections.

However, comparison of the cached and normal versions of the FMS showed only a small improvement in the overall performance of the system. This suggests that the overhead caused by creating many TCP connections is a minimal part of the communication process, and that although Microcosm messages are quite small, they do not result in a great deal of inefficiency when being sent across the network. For this reason, the socket-caching system was removed from the FMS.

8.4.4. Possible Improvements

Although the Docuverse extensions described above are sufficient to support the access required by the distributed FMS, it is currently deficient in the direct user interface offered. Although the user may browse the available local documents through a dialogue provided by the Docuverse, there is no way for them to access remote documents in this way.

To add this functionality, the Docuverse would need to offer a way for the user to select the particular host system whose Docuverse should be analysed. An extension of the Docuverse search facility would then allow the remote system to supply an appropriate list of documents.

In the current version of the system, a compromise solution is to use a shared Docuverse. Using the file sharing facilities of a LAN, a single Docuverse file can be maintained containing details of the documents available in the local system. This is accessed by the local Microcosm host systems, and as necessary files may be retrieved from the other systems. However, this is not a very satisfactory solution, since the user cannot always tell where a document is stored.

8.5. System Testing

8.5.1. Performance Issues

Once implemented, the performance of the distributed version of Microcosm, using various host systems, was evaluated. This was done by comparing the system response of local and remote filters. The linkbase used in the similar experiments described in chapters six and seven was installed on various systems, and the time taken to perform equivalent tasks was recorded.

The graph in figure 8.4 below illustrates the performance achieved by the various systems with the linkbase being accessed locally. The relative performance of each system is in keeping with their respective capabilities, the 486 processors being grouped closely, whilst the 386 system is significantly slower.

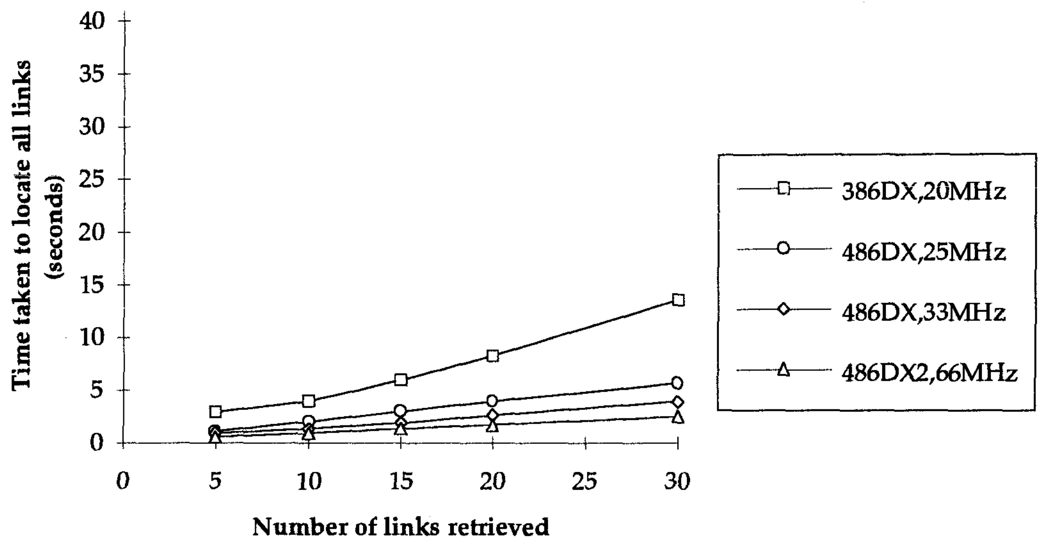


Figure 8.4 : The comparative performance of a variety of different host systems, each accessing a Microcosm linkbase locally.

The next graph, shown in figure 8.5, illustrates the equivalent results obtained when each of these host systems is used as a Microcosm server, providing a published linkbase for a remote client to access. In each instance, the client was a 25 MHz 486DX system. As might be expected, the graph has characteristics similar to the local linkbase, although the actual time taken is greater due to the overhead incurred by sending messages via the network.

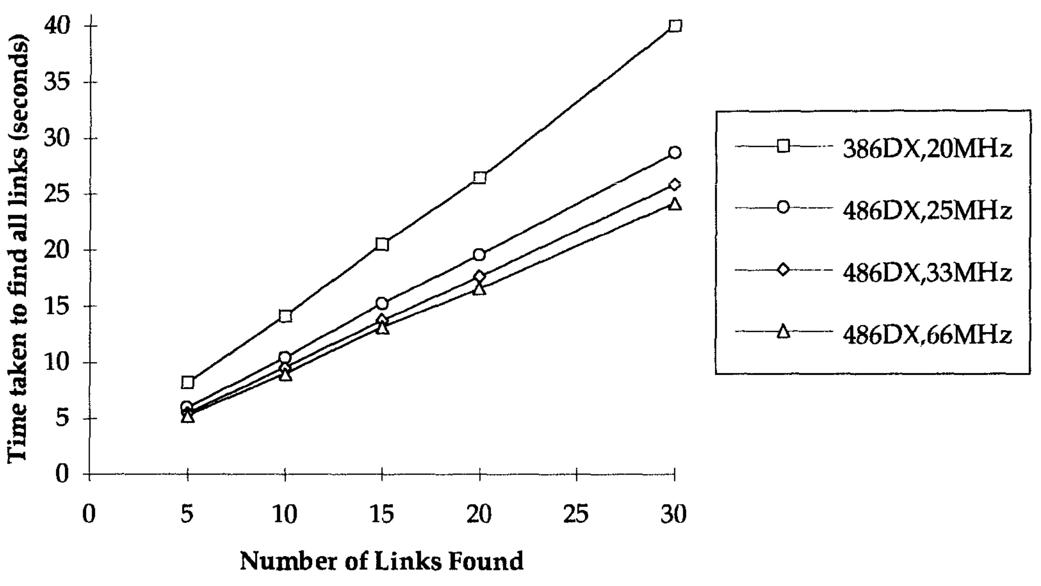


Figure 8.5 : The performance of various host systems when acting as a linkbase server to a remote Microcosm system.

The final graph (figure 8.6 below) shows the performance achieved when the test systems are used as Microcosm clients, accessing a linkbase from a remote server system. In this experiment, the server is a 25 MHz 486 DX. Although the response seen in the graph is still linear, the performance achieved by the various systems is grouped more tightly. This suggests that the performance in this instance is being bound by the performance of the server system, i.e. the 386DX/20 appears to be closer to the other systems, since the difference between the network performance of the systems is less than the difference between their local processing capabilities. This does not however mean that the client performance is unimportant, since the 66MHz 486 system is still able to provide a clearly greater performance.

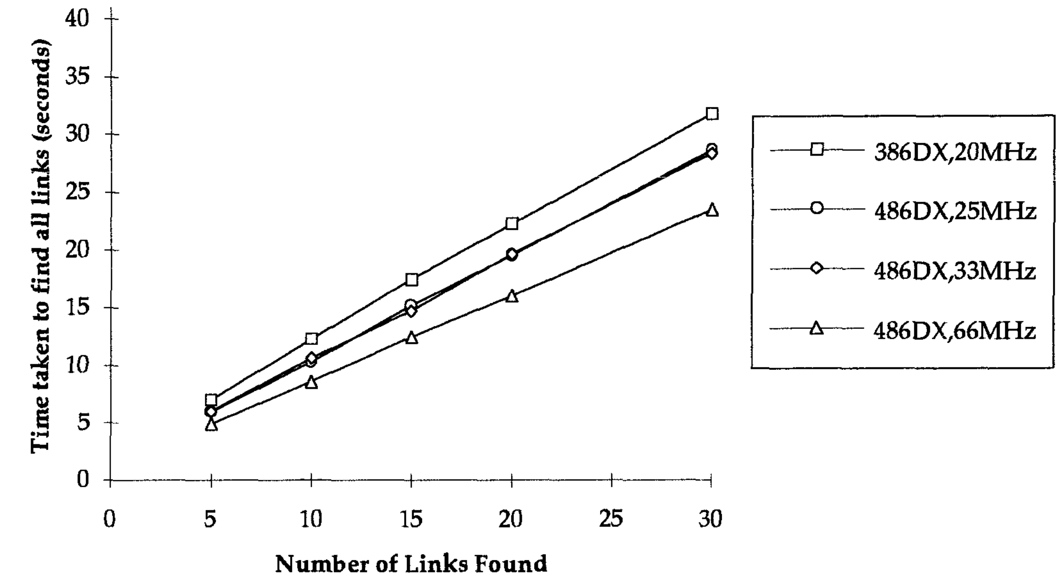


Figure 8.6 : The performance of various host systems when acting as a Microcosm client, accessing a linkbase from a remote server.

These experiments serve to show the type of performance that one can expect when using the distributed version of Microcosm. Obviously the precise configuration, and the particular filters being utilised will affect the actual performance attained. For example, processing time at the server will be reflected in an increase in overall response time, so the type of system chosen as a server, and the demands of the filters which it publishes, will affect the actual performance of the composite system.

Similarly, the number of users of a particular server (and the intensity of their use) will affect the performance which the server is able to offer. A more powerful system would also be able to handle the networking aspects more competently and therefore, depending on the actual situation, careful selection of server systems could greatly improve the performance possible. For a significant performance improvement, a system based upon a more powerful operating system (e.g. Windows NT or UNIX) would offer a good solution and this type of heterogeneous system will be investigated once the UNIX version of Microcosm reaches a suitable level of development.

Another aspect of perceived performance is the effect that carrying out distributed messaging has upon the rest of the local environment. Although there is no reliable metric for this type of performance, the normal environment was able to be used with no discernible reduction in response. Timing the Microcosm response whilst also interacting with other applications indicated that the networked response was reduced slightly. However, if the user is interacting with other applications then they are less likely to be as concerned with the performance that Microcosm achieves. Similar observations were made when using a system as a server, although a more processor-intensive filter would have a more noticeable affect on performance.

The final performance area is that of the distributed Docuverse system. Obviously the key factor in this case is the physical size of the file being retrieved. Careful choice of file formats where possible will help improve performance (e.g. using a compressed image format like JPEG rather than a less efficient format such as Windows bitmaps). However, a progress window is used to provide a guide to the transfer rate being achieved. Again, the rest of the Windows environment can be used during the transfer, but at the expense of the transfer rate. It is also possible to carry out more than one transfer at the same time, but this reduces the amount of processor time that can be utilised by each file transfer.

8.5.2. Heterogeneous Interactions

Although the only current implementation of the distributed Microcosm system is for the Windows environment, the use of TCP/IP as the basis for the networking functions means that the creation of versions for alternative platforms such as Apple Macintosh and X-Windows is simply a matter of adhering to the appropriate protocols. Versions of Microcosm for these platforms are currently under development (Lewis, 1993; Melly, 1993), and the incorporation of the distributed services described in this chapter may be investigated once they reach a more mature state.

However, it is the nature of open protocols that they are not restricted to a closed set of conforming applications. To demonstrate this, some experiments have been carried out with a UNIX-based distributed Lisp system (De Roure, 1990) to show how an application operating in a different environment, and developed using a very different language can interoperate with Microcosm. These experiments have so far led to the development of routines to support short interactions with Microcosm servers, retrieving details of published filters, and exchanging Microcosm messages. This has demonstrated the open features of the distributed messaging protocol used by Microcosm.

8.6. Summary

The distributed Microcosm model described in this section can clearly be seen to offer a greater flexibility of configuration than the systems described in chapter four, which are based upon a much more rigid client-server architecture. Additionally, the way in which this flexible configuration is made possible, through the utilisation of the modular structure of the back-end layer of Microcosm, allows the benefits of extensible hypermedia functionality which were identified in chapter six to be applied in a distributed environment.

The extension of the Microcosm Docuverse which has taken place to accommodate the distributed version of the FMS also brings many other advantages to the system. By providing a framework within which documents from other distributed information servers may be incorporated, it is possible to incorporate these services seamlessly within the system. For example, a filter designed to carry out WAIS queries would be able to form its results in such a way that they could be presented to the user via the Available Links viewer, and when a document found in this way is selected, the appropriate viewer is able to transparently access the remote WAIS server through the QD_OpenFile Docuverse function.

Finally, by avoiding the use of complicated protocols, it is very simple to allow other systems to interact with Microcosm through its network extensions.

9. Future Work

As described in the previous chapters, one of the original aims behind the development of Microcosm was to provide a basis for a wide range of research into the development of hypermedia systems. As can be seen from the work described in this thesis, and the range of other Microcosm-based work which has been referenced, the system has succeeded in this. In addition to the work described previously, this chapter describes some of the possible research directions which have been opened by the developments described earlier.

9.1. Development based on the Active FMS

The discussion in chapter seven clearly shows the advantages to be gained from the use of an active Filter Management System. As a result of the increased flexibility which this approach brings, many other possible enhancements to the functionality of the system become feasible.

9.1.1. *Additional Automation of Filter Configuration*

At present, the limitations of the chain implementation as described in section 6.3.3 still exist to a lesser extent in the filter sub-chains, which are created in the active system for each action offered by the set of executing filters. Although the active FMS provides a great deal of improvement in the efficiency of the system, there could still be a problem in ensuring that filters are ordered correctly within the various sub-chains.

In the active FMS, the problem of ordering of filters relative to each other, in order to handle the creation and receipt of particular message types, is taken care of by the action-based routing of messages. For example, the 'Dispatch' messages created by Linkbases are routed to the 'Dispatch' action sub-chain. However, if there are ordering requirements relating to two filters that process the same action, these filters will be located within the same sub-chain and the ordering must be handled by the user as before. For example, a filter might wish to analyse the links found by the system and use some form of heuristic to discard links that do not relate to the user's current task. This filter would register with the FMS to receive 'Dispatch' messages, but in order to ensure that the user would not see the links that it chooses to remove, it must be located before the Available Links filter, or its equivalent, in the 'Dispatch' action sub-chain.

It may be possible to refine the current implementation in order to ease the problem of ordering filters further. To do this a way of classifying the ordering requirements of a filter is required, which can vary according to the particular function a filter provides. For example, it doesn't matter what order Linkbases are placed in within the 'Follow Link' sub-chain, as 'Follow Link' messages will pass through them all, and any resulting links are simply passed on to the 'Dispatch' sub-chain. However, in the 'Dispatch' sub-chain, order can become very important, such as in the example above.

Although it might be possible to sub-class the various filters definitively according to behaviour, it is extremely unlikely that all possibilities can be incorporated satisfactorily due to the flexibility that is available to filter developers. The most workable solution would appear to be to allow filters to specify their ordering requirements quite broadly (e.g. beginning/middle/end/don't care). This information can be used as a first pass at placing the filter. If there are no filters with the same requirement, this pass is sufficient to place the filter correctly relative to the other filters in a particular action chain. Eventually however, a situation will occur where there is a clash of requirements between filters; for example, a new filter may wish to be placed towards the beginning of a sub-chain in which other filters with this requirement already exist. In this situation, the system may ask the user to specify their desired placement for the filter, relative to the filters which it clashes with. Further refinements of this approach could allow the user's chosen placement to be stored as a guide should the same situation occur again.

9.1.2. Automatic Menus

Although the dynamic menu library described in section 7.2.3 works satisfactorily, it has not been evaluated fully and there are a number of concerns over the way users might perceive menus which change as the filter configuration is altered. For example, if an individual uses Microcosm in more than one location, he or she may see a subtly different menu interface in each case due to the particular configurations of these systems. This loses the advantage of interface consistency that is an important aspect of graphical interfaces such as Windows. Alternatively, a naive user may alter their filter configuration only to become confused when this results in changes to the menus that they access.

Since the library that provides this dynamic menu support is a separate, shared resource, it is possible to experiment with solutions to this problem by modifying its functionality, and there are a number of options that have been suggested as solutions to this problem. One possibility is that a set of 'core' Microcosm actions could be defined (e.g. 'Follow Link', 'Start Link') which will always be present in menus, in a consistent position. When these actions are not catered for by the current configuration, the menu system will provide menus with these items present but disabled. Only less common, or specialised, actions will appear and disappear, and only from their own section of the menu. Another possibility is for the user, or a system administrator, to provide menu templates with their own core actions defined, thus retaining the maximum amount of configurability for the system. Issues such as these are addressed by Lewis in his work in the provision of hypermedia functionality as a system level service (Lewis, 1993), and are also being actively addressed in the current development of the Windows version of Microcosm.

9.1.3. Access to Filter Configuration Details

Another aspect which can be investigated further is increasing the level of access to the information that the active FMS stores about the available filters. If other components of the system were able to enquire about the actions that are available, they could avoid sending unnecessary messages that will not be processed, and if necessary, could start any filters that they specifically require, having first checked to see if the filter is already installed.

9.1.4. Improved Message Passing

There is also scope for improvement in the passing of messages between components of Microcosm, in particular between the FMS and filters where the bulk of message transfers occur. One such possibility is that groups of messages to be sent by a particular system element could be batched together and sent in one transfer. This would reduce the overhead involved in actually delivering the messages. For example, a Linkbase filter could group all 'Dispatch' messages being returned in response to a 'Follow Link' message into a single package which would in turn be passed by the FMS to the Available Links filter or similar. The receiving filter could then unpack the various messages in order to process them. This approach could be of particular benefit for distributed operation, where the overhead required to package information for transfer across the network would be reduced.

9.2. Extensions to the Distributed Microcosm system

The distributed version of Microcosm described in the previous chapter is very much a starting point for further investigation of the issues involved in the creation of flexible, distributed hypermedia systems. Like the single-user version of Microcosm, the distributed system offers a modular framework for such research.

Some of these areas, for example the integration of other distributed information systems, have been described in chapter four. The following sections outline some other potential areas of research.

9.2.1. Refining the Distributed Filter Model

In the existing implementation of the distributed FMS, the only level at which information may be made available is as a single published filter. Although this allows the maximum range of flexibility, it is a very fine level of granularity should the system be exmployed widely.

In order to accomodate the requirements of distributing diverse filters across a wide area, the system could be refined to allow groups of filters to be made available as a single logical entity. For example a large server might offer linking facilities for several different subjects, the various filters for each subject could be grouped to make access easier.

An additional possibility is the provision of indirect access to published filters. This would allow users to provide access to interesting facilities that have been discovered, or the development of 'composite' servers which bring together related filters from a variety of sources. This could be particularly useful in conjunction with the publication of filter groups a described above, allowing a user to add further information to a filter group and then making the new group available.

9.2.2. Computer Supported Collaborative Work

The use of hypermedia principles as an element of Computer Supported Cooperative Work (CSCW) systems has been discussed by many authors (e.g. Streitz et al., 1992; Malcolm et al., 1991; Burger et al., 1991) and investigation into the use of Microcosm to provide such a system is currently taking place (Melly, 1993).

In order to support the requirements of such a system, a number of extensions to the existing facilities of Microcosm are required. In particular, although the Microcosm Docuverse understands the concept of document authors, it does not offer any control over access to documents. In order to prevent authoring clashes in a cooperative system, the Docuverse must be extended to provide some form of locking. Other work in this area has identified approaches to concurrent document access that are suited to CSCW systems (Wiil and Leggett, 1993; Gronbaek et al., 1993), in particular the use of persistent, fine-grained locking and notification. In addition, tools to allow communication between collaborators would be required, such as shared whiteboard systems, and audio/video communication

With suitable extensions to the Docuverse system, and the development of appropriate specialised tools, the distributed Microcosm system would offer an ideal platform for investigating such systems, using the modularity of the FMS to add specific functions, and utilising the distributed facilities to share them.

9.2.3. New Network Facilities

With the current increase in interest in delivering multimedia information via a network (Adie, 1993), and the associated growth in world-wide connectivity, existing networking hardware and protocols are becoming outdated, unable to support the required bandwidth.

The distributed version of Microcosm is ideally suited as a platform for the investigation of new network facilities as they become available. Some of the areas that must be addressed for true distributed multimedia support are outlined below.

9.2.3.1. High Performance Networking.

As high performance networks such as SuperJANET and other ATM[1]-based networks become more widespread, a need for protocols suitable for the delivery of bandwidth-intensive multimedia information (e.g. video, audio) will become more pressing. These protocols must be able to deal with the time-critical aspect of such information and the need to maintain 'Quality of Service' to clients. This requires protocols to be developed which are able to compromise in order to meet an application's particular requirements (Little and Ghafoor, 1990). For example, a protocol able to automatically drop frames when delivering video, in order to match the timing requirements of the particular sequence with the capacity of the network at a particular time. As a flexible hypermedia framework with support for the modular configuration of network access, the distributed version of Microcosm is well suited as a framework for the investigation of such protocols.

9.2.3.2. Support for Distributed Applications

As well as advances in the underlying networks such as those described above, there are also new systems becoming available to support the development and operation of distributed applications.

These systems typically provide a high level interface to the problem of locating network services, and the interoperability between such services. Often known as Distributed Object Management Systems (DOMS), they allow a process to be encapsulated within an object which handles all network interactions on the process's behalf. This object is provided with all necessary information about the process (the functionality it provides and the interface through which it may be accessed). Similarly, client processes are structured in this way, to allow them to access server objects.

Because all processes are abstracted from the actual network through the DOMS, network location and hardware platform is transparent to the process, and is taken care of by the DOMS. The DOMS offers services to processes to allow them to locate other objects and their attributes, and to allow existing applications to be

---

[1] Asynchronous Transfer Mode

encapsulated within the system. One such system is the Common Object Request Broker Architecture (CORBA) developed by the Object Management Group, a consortium of system vendors (Osher, 1992).

Another new network facility is provided by the Distributed Computing Environment (DCE). Based upon the remote procedure call paradigm, DCE is supported by the Open Software Foundation (OSF). Heterogeneity is provided through the use of a standard data representation, and the location of remote procedures can be carried out through the use of a name server. An RPC daemon on a well known port number is required for each server to inform calling clients of the appropriate port for the available procedures. DCE also allows the use of callback facility to provide authentication of clients through a variety of supported security options.

A notable feature of the DCE system is the use of threads which can provide better performance than the use of child processes. DCE is likely to become increasingly widespread as the members of OSF promote its use. Further detail on the DCE protocol can be found in (Renaud, 1993).

The use of application architectures such as these in conjunction with the distributed Microcosm model would allow further investigation into issues such as wide-area networking, location transparency, and replication of services. In particular, they would be able to provide security and authorisation facilities, which are lacking in the current implementation of Microcosm. This would enable the restricted publishing of filters and documents to specified users or groups only.

9.3. Future Open Hypermedia Issues

There are many issues that remain to be addressed in the development of open hypermedia systems. The discussion of existing systems in chapter three, and the description of Microcosm, show that the requirement to provide a standard for the interchange of information between heterogeneous systems is yet to be addressed. Although there are standards such as HTML and MHEG that could be utilised, there are also many issues that arise when translating between different hypermedia systems (Leggett and Schnase, 1994). For example, handling links with multiple destinations, and conflicts between underlying models (e.g. systems with embedded links versus those with separate link databases). Any interchange format must be flexible enough to incorporate any such differences.

Another issue that is becoming increasingly important as hypermedia systems are applied to larger bodies of information is the provision of support for versioning. This is a very difficult problem, since it can apply at a number of levels, from links and documents to complete structures, all of which are intimately related. These difficulties are increased if the system supports cooperative authoring, and appropriate support for such authoring is an important first step for the provision of version control. Existing research into versioning has highlighted these difficulties, and has suggested the adaptation of techniques from software engineering (Østerbye, 1992), and the importance of using composite structures as an essential element of version management (Haake, 1992).

Addressing these issues will become more important as attempts are made to make wider use of open hypermedia systems.

10. Conclusions

10.1. The Requirement for Open Hypermedia Systems

The many hypermedia systems that have been developed in recent years have demonstrated that the hypermedia approach can provide a very powerful mechanism for the structuring of information, and its subsequent access.

The systems which have seen most widespread use are closed (e.g. Guide, Toolbook), but have proved to be satisfactory when dealing with well defined bodies of information that are relatively static in nature e.g. educational material or on-line documentation. Additionally, the scripting facilities of such systems allow subtle tailoring of interfaces to suit particular information types or structures.

The main drawback of such systems is the proprietary format, and the dependence on embedded linking, which result in dependence on the tools provided by the supplier. A further problem is that the ability to extend the material to incorporate notes and additional links is limited. This sort of facility extends the utility of static material, and makes possible the use of hypermedia as a supporting technology for many other, more dynamic tasks (e.g. software engineering (Creech et al., 1991) and information structuring (Marshall et al., 1991)).

To allow wider and more flexible application of hypermedia techniques, open hypermedia systems have been proposed. As described in chapter two, these systems are typified by linking mechanisms which are external to the actual information being linked, a modular and extensible architecture, and the ability to operate on a network. Such systems are a significant step towards the all encompassing hypermedia-based environments described by visionaries such as Bush, Nelson and Van Dam. The advent of the personal computer in recent years has changed the way that computers are used, and has led to an increase in 'end-user computing' (Maclean et al., 1993) rather than systems based on limited, centralised facilities. The powerful applications that are available to users of desktop computers can only benefit from the additional functions that an open hypermedia service would provide.

The Microcosm system described in this thesis is explicitly designed to meet these requirements, in particular the modular architecture which allows it to be used as a basis for research into the various issues raised by such systems. The particular strengths of the system are the ability to incorporate existing applications with a minimal amount of effort, and the modular filter system. The latter has many advantages over the fixed functionality of most systems, for both developer and user:

- New functionality can be added very easily, without the need to change or rebuild the rest of the system. This is particularly useful in a research environment where the central Microcosm framework is able to provide a stable, but flexible framework for the investigation of hypermedia issues.

- The modular structure makes a wide range of configurations are possible, according to the requirements of a particular set of information or the current user of the system. For example, a particular application might utilise a specially developed filter set, and each user may utilise a private Linkbase to store personal links, and choose their favoured navigation aids.

- The configuration of the system may be updated dynamically during use to vary the functionality available. For example, to utilise alternative link sets, linking methods or navigation tools.

The initial implementation of the Filter Management System, which is responsible for providing this architecture, allowed these advantages to be demonstrated, and offered an initial platform for the development of hypermedia functionality in the form of filter modules. There are however, a number of disadvantages to the simple chain model used by this system, as illustrated by the experiments described in chapter six.

The active FMS was developed to address these limitations, and offers a much more efficient approach to the task of managing the installed filters. The experimental results presented in chapter seven show how the performance of the new system is not affected by complex filter configurations, and the improved interface makes the system less intrusive to the rest of the Windows environment. The second advantage of the active FMS is the support it can give to the distributed version of Microcosm, with its ability to address each filter individually.

10.2. The Provision of Distributed Hypermedia Facilities

In recent years, there has been a significant increase in the amount of information available on-line, via computer networks. In particular, the Internet network has seen significant growth from its early beginnings, when it was occupied predominantly by computer researchers at academic institutions and large computer companies. It now includes many other parties who are simply users of the system, rather than interested parties. These include non-computing academic departments, and individuals accessing the network from home. The resultant explosion of available information has largely been led by the development of so-called 'resource discovery systems', which make the available information more accessible to the user.

The management of such large information resources is the sort of application that hypermedia systems are often claimed to be suited to, and one of the most successful resource discovery systems is the World-Wide Web hypermedia system developed at CERN in Switzerland. Its main advantage over other such systems is its incorporation of other information systems, and the ability to provide a more flexible structuring of information; hypermedia links can occur throughout the available information, while other systems (e.g. Gopher) provide a much coarser, hierarchical structure.

The World-Wide Web however is a closed hypermedia system by the definition given in chapter two, since it relies upon the use of a mark-up format, and the hypermedia structure is therefore very static. Similarly, many other distributed hypermedia systems fail to meet all the requirements of this definition. A common trend in these systems is the use of strict client-server systems and commercial databases; this leads to an equivalent of the proprietary format problem, and provides little flexibility in system configuration for the individual user. In addition, most systems (Hyper-G and VNS excepted) are oriented towards the provision of a single master server, a solution which does not scale well to a more widely distributed environment.

The distributed Microcosm model, described in chapter eight, is designed to provide a more flexible approach to the distribution of hypermedia functionality, in line with the flexibility offered by the single-user Microcosm system. Rather than offer a strictly enforced architecture, the distributed system utilises the access to the individual filters provided by the active FMS to provide a basic model which offers a much finer granularity of function. By allowing any single filter to be 'published' for other systems to use, a wide selection of distributed configurations is possible. The possible configurations range from simple cooperation between users to the provision of all hypermedia functionality from a Microcosm system acting as a server.

The extension of the Microcosm document database, or Docuverse, to allow the retrieval of remote files provides complimentary functionality to the distributed FMS. The modular structure of its network access offers similar experimental advantages to the filter system, making possible the clean integration of other distributed information services, and also for the use of advanced network protocols to support demanding media such as digital video.

In effect, the information servers that are currently in widespread suffer from disadvantages which are analogous to those of closed hypermedia systems, as described in chapter two. A great deal of effort is involved in building the information structure, and updating or maintaining this information can be complicated and time-consuming. The distributed version of Microcosm that has been described offers the advantages of a true open hypermedia system offering separate linking facilities and the ability to incorporate a wide range of existing services. By using the facilities of Microcosm, the value of such systems may be greatly enhanced through the use of cross-service linking and personal annotation.

10.3. An Integrated Information Environment

With the advent of open systems, the use of hypermedia systems as agents for total integration of information, as they are often advocated (Malcolm et al., 1991; Davis et al., 1992b), is beginning to become a possibility. The key to such a system is the ability to act as a 'Link Service' to all aspects of a user's environment (e.g. their favourite word-processor, spreadsheet, or database), and to offer a flexible set of underlying hypermedia functions. With the addition of distributed functionality, the hypermedia system would also be able to support co-operation between colleagues, and the incorporation of a wide range of remote information resources.

This thesis has described how facilities such as these may be provided by the flexible architecture of Microcosm. As well as the provision of hypermedia functions, the extensible interaction based architecture could also be used to provide other system level extensions to the user's environment. For example, a variety of image processing techniques could be implemented as filters, and Microcosm compatible viewers used to create and manipulate images in conjunction with these filters.

Whilst such systems have the potential for significant improvements to the user's environment, the real place for such functionality is within the operating system. With a modular framework available at this level, application developers can be encouraged to utilise it in order to improve the quality of their products. There is the potential for a move from arrays of large, monolithic applications as are seen today in the ubiquitous IBM-compatible PC to a system which acts as a supporting framework for a range of compatible tools. For example, a user could choose a word-processing tool, but rather than receive a huge bundle of supporting tools such as spelling and grammar checkers, a more suitable alternative could be selected from any supplier and installed, in the system framework.

An environment such as this would have advantages for vendors, who would no longer be forced to develop and support large applications, and for users, who would be able to create a personalised environment, better able to meet their specific requirements.

A. Distributed Message Formats

A.1 Introduction

This appendix describes the protocol used to exchange information between distributed Microcosm systems. The following sections list the information that must be provided in messages, and how these messages should be delivered.

A.2. Inter-FMS Messages

A.2.1. Locating Microcosm Servers

This message is sent to a pre-defined port (currently 1025) on a remote host, using the UDP protocol, in order to establish if the system supports a Microcosm system.

Query

Tag	Value
RUAServer	none required

Table A.1: Message to locate Microcosm servers

Reply

Tag	Value
McmServer	hostname

Table A.2: Message returning details of a remote Microcosm Server

A.2.2. Obtaining Details of Published Filters

Once a remote Microcosm system is located, a message can be sent to it requesting details of any filters which are published. This message is again sent using the UDP protocol to port 1025.

Query

Tag	Value
AnyPublishedFilters	none required

Table A.3: Request for details of published filters

Reply

Tag	Value
NumPublishedFilters	Number (N) of published filters
PublishedFilter1	Description
PublishedFilter2	Description
PublishedFilterN	Description

Table A.4: Details of published filters from a particular Microcosm server

A.2.3. Connecting to a Remote Filter

This message is used to 'connect' to a published filter, as identified using the messages described previously. Again, this message is delivered using the UDP protocol on port 1025. The reply contains all details required about the services offered by the filter.

Query

Tag	Value
ConnectToFilter	Filter Description
MicrocosmServer	Hostname

Table A.5: Message requesting connection to remote filter.

Reply

Tag	Value
Filter	Description
MCMServer	Hostname
NumActions	Number (N) of Actions
Action1	Action Details (<action>, <menu action?>, <menu string>, <applicable types>)
Action2	Action details as above
ActionN	Action details as above

Table A.6: Details of a particular filter to which a connection has been made

A.2.4. Transferring Microcosm Messages

The following message details are added to normal Microcosm messages being transferred between Microcosm systems. They allow the message and any additional messages generated to be returned to the source system. These messages are delivered using the TCP protocol to a server socket on port 1026. This protocol is more reliable, and allows a larger message to be delivered if necessary.

Tag	Value
MCMServer	Hostname of message source system
Filter	Description of Source/Destination filter

Table A.7: Additional tags used to route Microcosm messages between distributed systems

A.3. Inter-Docuverse Messages

Messages are exchanged between Microcosm Docuverses using the TCP protocol, on its own network port, 1027. The transaction is started with a request describing the action required, and the appropriate results are returned.

Query

Tag	Value
Request	Type of action required (FetchDocument, LookupDocument)
DocumentID	URL format Microcosm document identifier

Table A.8: Format of remote request to the Microcosm Docuverse

The result of these queries varies depending on the particular action. The LookupDocument action returns the Docuverse record for the document if available. The FetchDocument action will typically return more information, therefore, in reply to such a query the size of the document is first returned (to allow a progress indicator to be implemented) and then the actual document data.

Bibliography

ADIE, C., Network Access to Multimedia Information, Report by Réseaux Associés pour la Recherche Européenne (RARE), 1993, available in various formats by anonymous ftp from ftp.edinburgh.ac.uk in the directory /pub/mmaccess.

AKSCYN, R.M., MCCRACKEN, D.L., YODER, E.A., KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations, Communications of the ACM 31(7), July 1988, 820-835.

ALBERTI, B., ANKLESARIA, F., LINDNER, P., McCAHILL, M. *and* TORREY, D., (1992a) The Internet Gopher Protocol: A Distributed Document Search and Retrieval Protocol. Available by anonymous ftp from boombox.micro.umn.edu as protocol.txt in directory pub/gopher/gopher_protocol

ALBERTI, B., ANKLESARIA, F., LINDNER, P., McCAHILL, M. *and* TORREY, D., (1992b) Gopher+: Proposed Enhancements to the Internet Gopher Protocol. Available by anonymous ftp from boombox.micro.umn.edu as gopher+.txt in directory pub/gopher/gopher+

APPLE COMPUTER INC., Macintosh Hypercard User's Guide, 1987.

BEITNER, N.D., Multimedia Support in Hypermedia, Mini-Thesis, Department of Electronics and Computer Science, University of Southampton, UK, July, 1993.

BERNERS-LEE, T., CAILLIAU, R., GROFF, J., POLLERMANN, B., World-Wide Web: The Information Universe. CERN, 1211 Geneva 23, Switzerland, 1992.

BERNERS-LEE, T., HTTP: A Protocol for Networked Information, (1993a) available by ftp as http-spec.ps from info.cern.ch in directory /pub/www/doc.

BERNERS-LEE, T., Hypertext Markup Language (HTML), (1993b) available by anonymous ftp as html-spec.ps from info.cern.ch in directory /pub/www/doc.

BERTRAND, F., COLAÏTIS, F., LÉGER, A., The MHEG Standard and its Relation with the Multimedia and Hypermedia Area. In Proceedings of the IEE Conference on Image Processing and its Application (Maastricht, Holland, 7-9 April 1992), 123-6.

BIEBER, M., Issues in Modelling a "Dynamic" Hypertext Interface, In: Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec. 15-18), ACM Press, 203-217.

BRAND, S., The Media Lab, Penguin Books, 1988, ISBN 0 14 00.9701 5.

BRIGGS, J.H., TOMPSETT, C. *and* OATES N. Using Rules to Guide Learners Through Hypertext, Computers and Education, 20(1), February 1993, UK, 105-110.

BROWN, P.J., Hypertext: The Way Forward, *In:* J.C. van Vliet ed., *Document Manipulation and Typography,* Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography, Nice, France, April 20-22, 1988, Cambridge University Press, 183-191.

BURGER, A.M., MEYER, B.D., JUNG, C.P., LONG, K.B., The Virtual Notebook System, In: Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec. 15-18), ACM Press, 395 - 402.

BUSH, V., As We May Think, Atlantic Monthly, July, 1945, 101 - 108.

CAMPBELL, B. *and* GOODMAN, J. M., HAM: A General Purpose Hypertext Abstract Machine, Communications of the ACM 31(7), July 1988, pp 856-861.

CARD, S.K., ROBERTSON, G.G. *and* MACKINLAY, J.D., The Information Visualizer, An Information Workspace, *In:* Proceedings of CHI '91, ACM Conference on Human Factors in Computing Systems (April, 1991, New Orleans), ACM, New York, 1991, 181-188.

CARR, R.M., The Point of the Pen, Byte, 16(2), February, 1991, 211-221.

CATLIN, K.S., GARRETT, L.N. *and* LAUNHARDT, J.A., Hypermedia Templates: An Author's Tool, *In:* Hypertext '91, Proceedings of the Thirrd ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 147-160.

COLSON, F. *and* HALL, W. Multimedia Teaching with Microcosm-HiDES: Viceroy Mountbatten and the Partition of India. History and Computing 3(2), 1991, 89-98.

COMER, D. E., Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architecture, Prentice-Hall International Editions, 1991.

CONKLIN, J., Hypertext: An Introduction and Survey, IEEE Computer, September 1987, pp 17-41.

CREECH, M.L., FREEZE, D.F. *and* GRISS, M.L. Using Hypertext in Selecting Reusable Software Components. *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 25 - 38.

DAVIS, H., HALL, W., HUTCHINGS, G., RUSH, D. *and* WILKINS, R., (1992a) Hypermedia and the Teaching of Computer Science: Evaluating an Open System. *In:* Proceedings of the First Conference on Developments in the Teaching of Computer Science. (University of Kent, UK, 1992).

DAVIS, H., HALL, W., HEATH, I., HILL, G. *and* WILKINS, R., (1992b) Towards an Integrated Information Environment with Open Hypermedia Systems. *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92*. Proceedings of the Fourth ACM Conference on Hypertext (Milan, Italy, November 30-December 4, 1992), ACM Press, 181-190.

DE ROURE, D.C., A Lisp Environment for Modelling Distributed Systems, Ph.D. Thesis, Department of Electronics and Computer Science, University of Southampton, UK, 1990.

DESLISLE, N. *and* SCHWARTZ, M., Neptune: A Hypertext System for CAD Applications. *In:* Proceedings of the ACM SIGMOD International conference on the Managment of Data, 132-143, 1986.

EASTGATE SYSTEMS, INC., Storyspace, Hypertext Writing Environment for Macintosh Computers, 1991.

EDWARDS, I., A Docuverse for Microcosm, 3rd Year Project Report, Department of Electronics and Computer Science, University of Southampton, UK, 1992.

EGAN, D.E., LESK, M.E., KETCHUM, R.D., LOCHBAUM, C.C., REMDE, J.R., LITTMAN, M. *and* LANDAUER, T.K., Hypertext for the Electronic Library? CORE Sample Results, *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press,1991, 299-311.

EGAN, D.E., REMDE, J.R., LANDAUER, T.K., LOCHBAUM, C.C. *and* GOMEZ, L.M., Behavioral Evaluation and Analysis of a Hypertext Browser, *In:* Proceedings of

CHI '89: Human Facors in Computing Systems (Austin, TX., April 30 - May 4, 1989), ACM Press, 205-210.

EMTAGE, A., DEUTSCH, P., Archie: An Electronic Directory Service for the Internet. In: Proceedings of the Winter 1992 Usenix Conference (San Francisco, California, 20-24 January 1992), 93-110.

ENGELBART., D.C. *and* ENGLISH, W.K., A Research Centre for Augmenting Human Intellect, AFIPS Conference Proceedings, 33(1), 1968, 395-410.

FALOUTSOS, F., LEE, R., PLAISANT, C. *and* SHNEIDERMAN, B., Incorporating String Search in a Hypertext System: User Interface and Signature File Design Issues, Hypermedia, 2(3), 1990, 183-200.

FERSKO-WEISS, H., 3-D Reading with the Hypertext Edge, PC Magazine, May 1991, 242-282.

FOUNTAIN, A., HALL, W., HEATH, I. *and* DAVIS, H., Microcosm: an Open Model With Dynamic Linking. *In:* A. Rizk, N. Strietz *and* J. André, Eds *Hypertext: Concepts, Systems and Applications.* Proceedings of the European Conference on Hypertext, (INRIA, France, November 1990), 298 - 311.

FURNAS, G.W., Generalised Fisheye Views, *In:* Proceedings CHI'86 Human Factors in Computer Systems, Boston, Massachusetts, April 13-19, 1986, ACM, New York, 16-23.

FURUTA, R., STOTTS, P. D., The Trellis Hypertext Reference Model. *In: Proceedings of the NIST Hypertext Standardisation Workshop,* NIST Special Publication SP500-178, 83-93.

GARZOTTO, F., PAOLINI, P. *and* SCHWABE, D., HDM - A Model for the Design of Hypertext Applictions, *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press,1991, 313-328.

GAVIOTIS, I., HATZIMANIKATIS, A. *and* CHRISTODOULAKIS, D., An Architecture for Active Hypertext on Distributed Systems. *In:* Proceedings of CompEuro '92: Computer Systems and Software Engineering (The Hague, The Netherlands, 4-8 May, 1992), IEEE Computing Society Press, USA, 377-382.

GLOOR, P.A., CYBERMAP - Yet Another Way of Navigating in Hyperspace. *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press,1991, 107 - 122.

GRØNBÆK, K. *and* TRIGG, R.H., Design Issues for a Dexter-Based Hypermedia System. *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92.* Proceedings of the Fourth ACM Conference on Hypertext (Milan, Italy, November 30-December 4, 1992), ACM Press, 191-200.

GRØNBÆK, K., HEM, J.A., MADSEN, O.L. *and* SLOTH, L., Designing Dexter-based Hypermedia Systems. *In:* Proceedings of the Fifth ACM Conference on Hypertext (Seattle, WA, November 14-18, 1993), ACM Press, 25-38.

HAAKE, A., CoVer: A Contextual Version Server for Hypertext Applications, *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92.* Proceedings of the Fourth ACM Conference on Hypertext (Milan, Italy, November 30-December 4, 1992), ACM Press, 43-52.

HAAN, B.J., KAHN, P., RILEY, V.A., COOMBS, J.H. *and* MEYROWITZ, N.K., IRIS Hypermedia Services, Communications of the ACM, (35, 1), 1992, 36-51.

HALASZ, F.G., Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. Communications of the ACM, (31,7), 1988, 836 - 852.

HALASZ, F.G. *and* SCHWARTZ, M., The Dexter Hypertext Reference Model. *In: Proceedings of the NIST Hypertext Standardisation Workshop,* NIST Special Publication SP500-178, 95-133.

HALASZ, F.G., Seven Issues: Revisited. Closing Keynote Address, Hypertext 91, San Antonio TX, 1991.

HALASZ, F.G., MORAN, T.P. *and* TRIGG, R.H., Notecards in a Nutshell, *In:* Proceedings of the 1987 Conference of Human Factors in Computer Systems (CHI + GI 87) (Toronto, Ontario, April 5 - 9, 1987), 42 - 52.

HALL, M., TOWFIQ, M., ARNOLD, G., TREADWELL, D., SANDERS, H., Windows Sockets, An Open Interface for Network Programming under Microsoft Windows, January, 1993. Available by anonymous ftp as winsock11.doc from microdyne.com in the directory /pub/winsock.

HAYES, I., Specification Case Studies, Prentice Hall International Series in Computer Science, Prentice Hall International, 1987.

HEATH, I., An Open Model for Hypermedia: Abstracting Links from Documents, Ph.D. Thesis, University of Southampton, UK, 1992.

HILL, G., WILKINS, R. *and* HALL, W., Open and Reconfigurable Hypermedia Systems: A Filter Based Model. Hypermedia, 5(2), 1993, 103-118.

HOLLOM, R., Integrating Internet Resource Discovery Services with Open Hypermedia Systems, Computer Science Technical Report CSTR 93-18, University Of Southampton, UK, 1993.

JACKSON, S. *and* YANKELOVICH, N., InterMail: A Prototype Hypermedia Mail System, *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 405-409.

KACMAR, C.J. *and* LEGGETT, J.J., PROXHY: A Process-Oriented Extensible Hypertext Architecture, ACM Transactions on Information Systems, 9(4), 1991, 399-419.

KAHLE, B., Wide Area Information Server Concepts, Thinking Machines, 1989.

KAPPE, F., MAURER, H. *and* SHERBAKOV, N., Hyper-G: A Universal Hypermedia System, March, 1992. Available by anonymous FTP from iicm.tu-graz.ac.at as report333.txt.Z in directory pub/Hyper-G/doc.

KAPPE, F., PANI, G., *and* SCHNABEL, F., The Architecture of a Massively Distributed Hypermedia System, Internet Research: Electronic Networking Applications and Policy, 3(1), Spring 1993, 10-24.

LEGGETT, J.J. *and* KILLOUGH, R.L., Issues in Hypertext Exchange. Hypermedia 3(3), 159-186.

LEGGETT, J.J. *and* SCHNASE, J.L., Viewing Dexter with Open Eyes, Communications of the ACM, 37(2), 1994, 77-86.

LESLIE, M., The Hartlib Papers Project: Text Retrieval with Large Datasets, Literary and Linguistic Computing, 5(1), 1990, 58-69.

LEWIS, A.J., A Hypermedia Link Service as an Operating System Extension, Mini-thesis, Department of Electronics and Computer Science, University of Southampton, UK, 1993.

LI, Z., HALL, W. *and* DAVIS, H., Hypermedia Links and Information Retrieval. In Proceedings of the 14th British Computer Society Research Colloquium on Information Retrieval (Lancaster University, U.K., 1992), Springer Verlag (1993), 169-180.

LITTLE, T. *and* GHAFOOR, A, Network Considerations for Distributed Multimedia Object Composition and Communication. IEEE Network, November 1990, 32 - 49.

LYNCH, C.A., The Z39-50 Information Retrieval Protocol: An overview and Status Report. ACM Computer Communication Review, 21(1), 1991, 58-70.

MALCOLM, K.C., POLTROCK, S.E. *and* SCHULER, D., Industrial Strength Hypermedia: Requirements for a Large Engineering Enterprise. *In:* Hypertext 91: Proceedings of Third ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 13 - 24.

MAROVAC, N. *and* OSBURN, L., HyperNet: A Tool to Choreograph Worldwide Distributed Hypermedia Documents, Computers and Graphics, 16(2), 1992, UK, 197-202.

MARSHALL, C.C., HALASZ, F.G., ROGERS, R.A., *and* JANSSEN, W.C., Aquanet: A Hypertext Tool to Hold Your Knowledge in Place, *In:* Hypertext '91, Proceedings of the 3rd ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 261-275.

MAURER, H. *and* TOMEK, I., (1990a) Some Aspects of Hypermedia Systems and their Treatment in Hyper-G, Wirtschaftsinformatik, 32(2), 1990, 187-196.

MAURER, H. *and* TOMEK, I., (1990b) Broadening the Scope of Hypermedia Principles. Hypermedia, 2(3), 1990, 201-220.

McLEAN, E.R., KAPPELMAN, L.A., THOMPSON, J.P., Converging End-User Computing and Corporate Computing, Communications of the ACM, 36(12), 1993, 79-91.

MELLY, M., CSCW in an Open, Distributed Hypermedia System, Mini-thesis, Department of Electronics and Computer Science, University of Southampton, UK, 1993.

MEYROWITZ, N., The Missing Link: Why We're All Doing Hypertext Wrong. *In:* Barrett, E. (ed), *The Society of Text.* MIT Press, Cambridge, Massachusetts, 1989, 107-114.

MEYROWITZ, N., Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Application Framework. *In: Proceedings of OOPSLA 86*, September 1986, 186 - 201.

NANARD, J. *and* NANARD, M., Using Structured Types to Incorporate Knowledge in Hypertext, *In:* Hypertext '91, Proceedings of the 3rd ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 329-343.

NELSON, T., Getting It Out of Our System, *In:* Information Retrieval: A Critical Review, G. Schechter, ed., Thomson Books, Washington, D.C., 1967, 191-210.

NELSON, T., Literary Machines 87.1, published by the author.

NEUMAN, B.C., Prospero: A Tool for Organising Internet Resources, Electronic Networking: Research, Applications and Policy, 2(1), Spring 1992, 30-37.

NEWCOMB, S.R., Explanatory cover Material for Section 7.2 of X3V1.8M/SD-7, *In:* Proceedings of NIST Hypertext Standardisation Workshop, January 16-18, National Institute of Standards and Technology, USA, 1990. NIST Special Publication SP500-178, 179-188.

NOLL, J. *and* SCACCHI, W., Integrating Diverse Information Repositories: A Distributed Hypertext Approach, IEEE Computer, 24(12), 38-45.

OSHER, H.M., Software Without Walls, Byte, 17(3), 1992, 122-128.

ØSTERBYE, K., Structural and Cognitive Problems in Providing Version Control for Hypertext, *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92.* Proceedings of the Fourth ACM Conference on Hypertext, Milan, Italy, November 30-December 4, 1992. ACM Press, 33-42.

OWL INTERNATIONAL INCORPORATED, Guide: Hypertext for the PC, 1987.

PARUNAK, H.V.D., Don't Link Me In: Set Based Hypermedia For Taxonomic Reasoning. *In:* Hypertext '91, Proceedings of the 3rd ACM Conference on Hypertext (San Antonio, TX., Dec 15-18), ACM Press, 1991, 233-242.

PEARL, A., Sun's Link Service: A Protocol for Open Linking. *In:* Hypertext '89 Proceedings (Pittsburgh PA, 1989), ACM Press, 137 - 146.

Proceedings of the NIST Hypertext Standardisation Workshop, January 16-18, National Institute of Standards and Technology, USA, 1990. NIST Special Publication SP500-178.

RADA, R., Hypertext: From Text To Expertext, McGraw-Hill, UK, 1991.

RENAUD, P.E., Introduction to Client/Server Systems: A Practical Guide for Systems Professionals, Wiley, 1993.

RILEY, V., An Interchange Format for Hypertext Systems : The Intermedia Model. *In:* Proceedings of the NIST Hypertext Standardisation Workshop, January 16-18, National Institute of Standards and Technology, USA, 1990. NIST Special Publication SP500-178, 213-222.

RIZK, A., *and* SAUTER, L., Multicard: An Open Hypermedia System. *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92.* Proceedings of the Fourth ACM Conference on Hypertext (Milan, Italy, November 30-December 4, 1992), ACM Press, 4-10.

SCHÜTT, H. A. *and* STREITZ, N. A., *HyperBase:* A Hypermedia Engine Based on a Relational Database Management System, *In: Hypertext: Concepts, Systems and Applications* (Proceedings of the European Conference on Hypertext, INRIA, France, November 1990) Rizk, A., Strietz, N. *and* André, J., Eds, 95-108.

SCHWARTZ, M.F. *and* TSIRIGOTIS, P.G., Experience with a Semantically Cognizant Internet White Pages Directory Tool. Journal of Internetworking Research and Experience, 2(1), March 1991, 23-50.

SHACKELFORD, D.E., SMITH, J.B. *and* SMITH, F.D., The Architecture and Implementation of a Distributed Hypermedia Storage System, *In:* Proceedings of the Fifth ACM Conference on Hypertext (Seattle, WA, November 14-18, 1993), ACM Press, 1-13.

SHIPMAN, F.M. III, CHANEY, R.J. *and* GORRY, G.A., Distributed Hypertext for Collaborative Research: The Virtual Notebook System. *In:* Hypertext '89 Proceedings (Pittsburgh PA, 1989), ACM Press, 129 - 136.

STREITZ, N., HAAKE, J., HANNEMANN, J., LEMKE, A., SCHULER, W., SCHÜTT, H. *and* THÜRING, M., SEPIA: a Cooperative Hypermedia Authoring Environment, *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92*. Proceedings of the Fourth ACM Conference on Hypertext (Milan, Italy, November 30-December 4, 1992), ACM Press, 11-22.

TRIGG, R., MORAN, T. P. *and* HALASZ, F. G., Tailorability in NoteCards. *In: Proceedings of Interact '87*, 2nd IFIP Conference on Human-Computer Interaction (Stuttgart, West Germany, August 1987), 723-728.

TRIGG, R. H., IRISH *and* P. M., Hypertext Habits: Experiences of Writers in Notecards. *In: Hypertext '87 Proceedings* (Chapel Hill, North Carolina, November 13-15). 89 - 108.

VAN DAM, A., "Hypertext '87 Keynote Address," Communications of the ACM, July 1988, 31(7), 887 - 895.

WIIL, U.K. *and* LEGGETT, J., Hyperform: Using Extensibility to Develop dynamic, Open and Distributed Hypertext Systems. *In:* D. Lucarella, J. Nanard, M. Nanard *and* P. Paolini, *eds ECHT '92*. Proceedings of the Fourth ACM Conference on Hypertext, (Milan, Italy, November 30-December 4, 1992), ACM Press, 251-261.

WIIL, U.K. *and* ØSTERBYE, K., Experiences with Hyperbase - A Multi-user Back-end for Hypertext Applications with Emphasis on Collaboration Support. Technical Report R 90-38, Department of Computer Science, The University of Aalborg, Denmark, October 1990.

WIIL, U.K. *and* LEGGETT, J., Concurrency Control in Collaborative Hypertext Systems, *In:* Proceedings of the Fifth ACM Conference on Hypertext (Seattle, WA, November 14-18, 1993), ACM Press, 14-24.

YANKELOVICH, N., HAAN, B.J., MEYROWITZ, N.K. *and* DRUCKER, St.M., (1988a) Intermedia - The Concept and Construction of a Seamless Information Environment, IEEE Computer, 21(1), 1988, 81-96.