UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

# UNIFYING DISTRIBUTED PROCESSING AND OPEN HYPERMEDIA THROUGH A HETEROGENEOUS COMMUNICATION MODEL

Stuart Goose, Jonathan Dale, Gary J. Hill,
David C. DeRoure and Wendy Hall

University of Southampton

Technical Report No. 95-6

November 1995

ISBN: 0-854-325-824

# Unifying Distributed Processing and Open Hypermedia through a Heterogeneous Communication Model

Stuart Goose, Jonathan Dale, Gary J. Hill, David C. DeRoure and Wendy Hall

Multimedia Research Group

Department of Electronics and Computer Science

University of Southampton

UK

{sg93r, jd94r, gjh, dder, wh}@ecs.soton.ac.uk

*A successful distributed open hypermedia system can be characterised by a scaleable architecture which is inherently distributed. While the architects of distributed hypermedia systems have addressed the issues of providing and retrieving distributed resources, they have often neglected to design systems with the inherent capability to exploit the distributed processing of this information. The research presented in this paper describes the construction and use of an open hypermedia system concerned equally with both of these facets.*

## 1. Introduction

It is now widely accepted that hypermedia functionality can perform a useful role as an underlying component, or "link service" of an information system. This type of system, which is closer to the original perception of hypermedia envisioned by Nelson [20], has been described by various authors [24,9,13], and the applicability of such an approach illustrated [17]. Our own work at Southampton with the Microcosm system [9,15] uses the following criteria as requirements for such a link service:

- *Size*. It should be possible to import new nodes, links, anchors and other hypermedia objects without any limitation upon the size of the objects or the maximum number of such objects that the system may contain.
- *Data Formats*. The system should allow the import and use of any data format, including temporal media.
- *Applications*. The system should allow any application to access the link service in order to participate in the hypermedia functionality.
- *Data Models*. The hypermedia system should not impose a single view of what constitutes a hypermedia data model, but should be configurable and extensible so that new hypermedia data models may be incorporated. It should thus be possible to inter-operate with external hypermedia systems and to exchange data with other external systems.
- *Platforms*. It should be possible to implement the system on distributed platforms.
- *Users*. The system must support multiple users, and allow each user to maintain their own private view of the objects in the system.

In general, Microcosm meets these requirements, but is weak in its support for distributed operation. Hill [16] describes initial experimentation with a networked version of Microcosm. This paper reports significant advances upon Hill's contribution through the provision of a distributed framework that exhibits the desirable qualities of scaleability and efficiency, above which an open hypermedia system is layered.

## 2. Flexible Distributed Operation

The Microcosm system is based upon a set of autonomous communicating processes, which together can provide the functionality required of a hypermedia service. The aim is to maximise the flexibility and extensibility of the system to allow each user to customise the facilities available.

Although a number of current hypermedia systems support distributed operation, few are truly open systems and are unable to provide the degree of flexibility offered by the Microcosm model. The majority of these systems are also strongly tied to strict client/server configurations, which restrict the degree with which users may build a system from diverse resources. With the widespread increase in the use of networks, such as the Internet, for information dissemination, contemporary systems must be flexible enough to incorporate resources from a wide range of sources.

Many existing systems, such as the World Wide Web (WWW) [3], although open in terms of the platforms and protocols supported and the ability to handle distributed operation, does not explicitly support an open hypermedia model. The WWW in particular suffers due to its reliance on the Hypertext Mark-up Language (HTML) [4] for the provision of linking. This prevents hypermedia navigation from taking place within documents stored in other formats. In addition, the lack of a separate link management subsystem forces the user to move continuously from server to server as they navigate through documents. Minimal consideration has been afforded to users wishing to incorporate their own hypermedia tools with this system.

The Hyper-G system [1] whilst superficially similar to the WWW, is an advance due to the separation of link information into link databases. This facilitates the application of links to a wider range of document formats, for example images or word-processor documents, in a common manner. However the Hyper-G server only offers a simple hypermedia model and does not provide general extension facilities.

Hyperform [29], goes further by providing a fully extensible hypermedia server, or *hyperbase*. This provides a set of object classes to allow the hypermedia services provided to be tailored to particular requirements, and facilities to integrate external tools into the system. The integration facilities suggest that inter-operation between Hyperform servers would also be possible, but this is not explicitly supported.

The Virtual Notebook System (VNS) [25,6], provides an open hypermedia architecture based on the use of underlying database routines, thus allowing the development of custom programs that utilise the hypermedia facilities. The system also supports interaction between different VNS servers, allowing workgroups to be supported by dedicated servers whilst still permitting the sharing of information between individual workgroups.

Although all these systems support some form of a distributed model, none currently provide the degree of flexibility which a distributed version of Microcosm could provide. As described previously, the aim of the initial distributed Microcosm model offers a system which allows finely granulated distribution of hypermedia functionality, utilising the modular nature of the Microcosm model to deliver maximum flexibility to users. These extensions to the Microcosm model allow hypermedia functionality to be shared amongst users.

The prototype system described by Hill addressed some weaknesses in the system architecture of the stand-alone version of Microcosm in order to allow efficient distributed operation, and demonstrated the feasibility of fine-grained distribution of services. However, there was neither explicit support for distributed processing nor provision for handling large, complex configurations.

The following sections trace the development of a generic communication layer for the support of distributed systems, and describe how it provides the foundation for a new version of the distributed Microcosm system. We believe that this system should offer greater scaleability, better process management and more efficient communication.

# 3. Heterogeneous Communication Model

Wilkins and Heath [30] propose a scaleable message passing solution, called the Direct Communication Model (DCM), above which Microcosm could be layered. It provides several mechanisms by which peer entities executing on a single machine may communicate with one another. Whilst retaining much of the flexibility enjoyed by previous communication architectures for Microcosm, the scaleable topology of the DCM provides a speed increase due to the more efficient routing of messages. This model has been adapted and enhanced to function effectively within a distributed heterogeneous multi-user environment.

## 3.1. Process addressing scheme

For direct communication to take place, one process must be able to uniquely identify or address the process to which it wishes to send a message. This rudimentary requirement gave rise to a novel process addressing scheme using a familiar metaphor borrowed from computer filing systems [12].

The process addressing scheme forms the foundation upon which the support for message passing between processes executing on different machines is based. The first major benefit of the addressing scheme is that it can be customised by the architects of the host system to reflect the context in which it operates. This means that a process can send a message identifying the intended recipients using terms that have meaning within that context. The second key benefit is the *wildcard* attribute, which aids a process in disseminating a message to a wider audience. By placing a wildcard in one or more positions within the destination process address, a message can be targeted at a specific community of processes.

For example, one host system may require all process registrations to conform to the following custom addressing template:

/ProcessName/UniqueProcessID/Document/Service

Using this template, two link database processes could each register two service entries in the following style:

/linkbase/152.78.64.64.13/lion.raw/follow.link
/linkbase/152.78.64.64.13/lion.raw/create.link
/linkbase/152.78.64.48.11/tiger.raw/follow.link
/linkbase/152.78.64.48.11/tiger.raw/create.link

A viewer process could then direct a message to all link database processes that service follow link requests by quoting the following destination process address, where asterisk represents a wildcard:

/linkbase/*/*/follow.link

## 3.2. Architecture of the HCM

One of the key principles of distributed computing is that any processing need not be performed locally, but on the most appropriate machine. It was decided that the design of the HCM would embody this principle from the outset.

The DCM is implemented as a shared library to which all participating processes dynamically link. As this solution is clearly inappropriate for processes wishing to pass messages across machine boundaries, the central module through which communication is directed needed to be a process in its own right. This process is referred to as a *router*.

The router (figure 1) acts as a bureau where processes, using the addressing scheme as the vehicle, can dynamically advertise and withdraw their services and also post messages to other registered service providers.

A *process manager* is also started with each session to govern load balancing across the network and the distributed invocation of processes. Through its user interface, processes can be remotely configured and managed.

For the router and participating processes to dynamically configure their network connections, a supporting piece of infrastructure proved necessary. A single *daemon* process to serve the local network domain[1], providing the sole fixed point of contact, helped achieve this.

## 3.3. Scaleable topology

The linear chain topology of Filter Management System (FMS) within Microcosm is not suitable for distributed operation. Due to the simple message routing algorithm employed, excessive communication overheads are incurred which would restrict the construction of a scaleable system

[1] A domain is purely an administrative measure for specifying a list of machines that HCM processes may execute on.

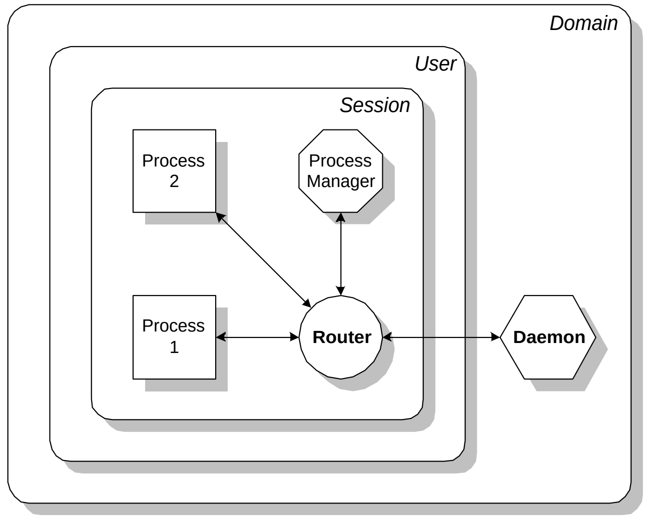


**Figure 1 : Logical Model of the HCM Architecture**

A crucial characteristic of a scaleable architecture is that the addition of processes to the system has minimal impact upon efficiency. Another vital scaleability issue specific to the communication architecture is that the number of intermediate steps travelled by a message remains constant regardless of the number of processes present within the system. Both of these properties are exhibited by the HCM; the latter can be appreciated in figure 2, as all of the processes are satellites of the router, resulting in a message only ever travelling two steps to its destination.

## 3.4. Insulation from network dependencies and complexities

For inter-process communication to occur between two distributed processes, one process must initiate the request by contacting the other. This requires that the initiating process is aware of the location of the remote process. Considering this scenario within the context of TCP/IP[1], the initiating process would have to know both the IP address of the machine where the remote process is executing and the port number on which it is listening for network activity. The customisable addressing mechanism mentioned above affords a degree of process location transparency. Using the HCM as a communication layer, the initiating process can then identify and target the services of a remote process in a meaningful context, as opposed to first determining and then supplying the network addresses of remote processes.

Processes interact with the HCM through an API[2], which was developed for a number of reasons. Many

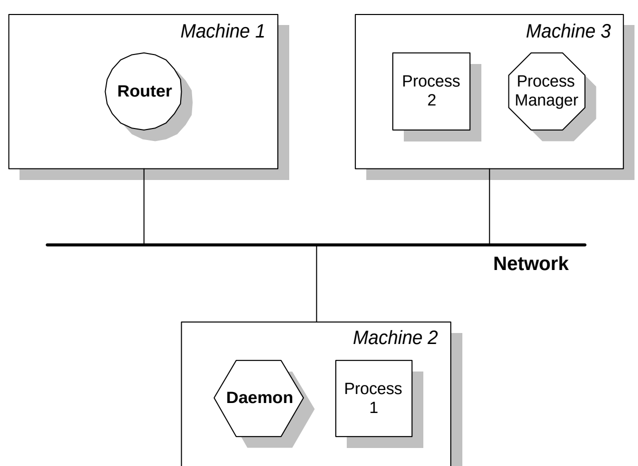


**Figure 2 : Physical Model of the HCM Architecture**

---

[1] The *de facto* networking protocol used by the Internet
[2] Application Programming Interface

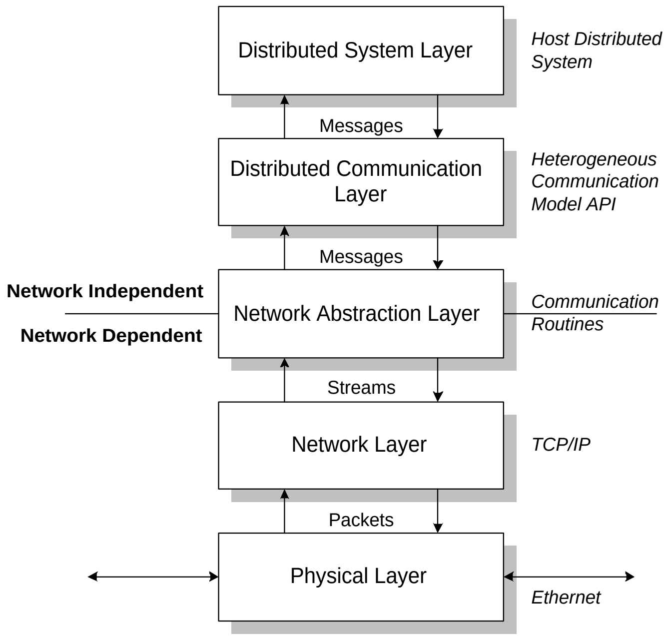


**Figure 3 : Layered HCM API**

application programmers would not be receptive when faced with the prospect of developing network aware HCM processes. The API insulates the programmer from explicitly managing any interaction with the network. The fundamental transport mechanism employed by the HCM is concealed beneath the Network Abstraction Layer [7]. The advantages of this layered approach are two fold: subsequent modifications and improvements can be made without affecting the modules that rely upon those services, and supporting another transport mechanism or protocol could be achieved transparently. The layered construction of the HCM API can be seen in figure 3.

## 3.5. Alternative configurations

The HCM can be configured as a:

- *Single user environment*.
- *Multi-user environment*, whereby processes belonging to one user may communicate with the processes of another user within a single domain.
- *Multi-user environment*, whereby processes belonging to one user may communicate with the processes of another user across widely distributed domain boundaries.

The logical view of the last configuration is presented in figure 4. User 1 in domain 1 initiates an action causing the daemon of domain 2 to be contacted, requesting a connection between their router and the router belonging to user 2. This currently means that the daemon in domain 1 must have previously obtained the network address of the machine in domain 2 where the daemon is located. Once a virtual connection has been established, negotiation between the two routers occurs. The process registration entries declared as *published* (that is, applications that are available to other users) are then exchanged between the two routers. This activity alleviates superfluous message passing between routers. The scaleable framework providing widely distributed, peer-to-peer communication can now be exploited.

The following two sections describe how this framework has been used as a foundation on which to build an open, distributed hypermedia system, known as Microcosm: The Next Generation (Microcosm TNG).

# 4. Heterogeneous User Interfaces

When a process is designed for a single processor machine, it is generally written with a specific platform and user interface environment in mind, for example, a Silicon Graphics ELF executable, a PC

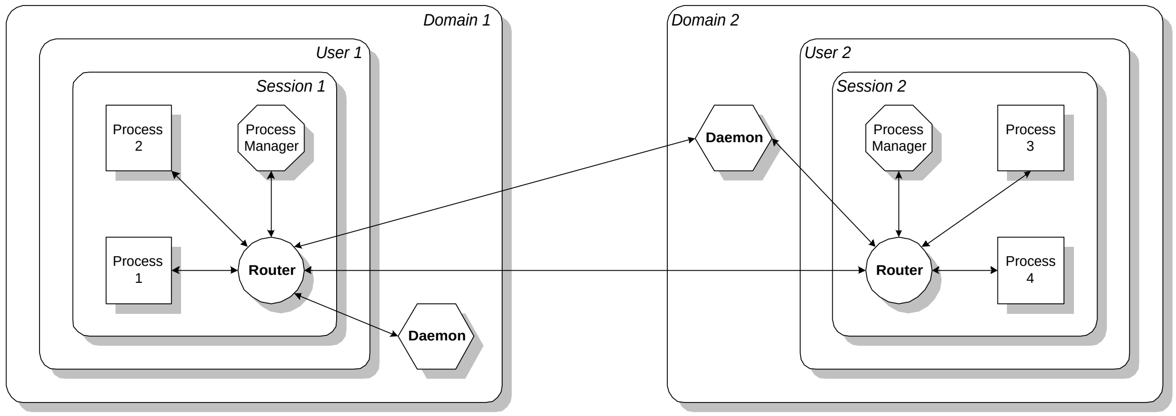


**Figure 4 : Widely Distributed Multi-user Environment**

Windows executable, etc. Generally, these types of processes are not written with any form of distribution in mind, so the code to handle the user interface of a process and the code to handle the functionality of a process are written together, side-by-side.

Yet, even if good programming practices are employed to ensure that the user interface environment code remains separate from the functionality code of a process, this still means that the user interface part of a process must be rewritten for each user interface environment (Windows, X-Windows, Macintosh, etc.). It would be desirable, therefore, if a process could retain a portable functionality segment and adopt a user interface environment segment dependent upon the user interface environment it was to be executed within. This would allow the process to execute on a wide range of machines within the distributed environment.

The X Windows system [21] attempts to alleviate this problem somewhat by allowing user interfaces to be redirected over the network to other X-supporting servers. A more heterogeneous and wider solution is required to allow all of the user interfaces belonging to a specific user to be redirected to their display regardless of the user interface environments within the distributed environment.

To give a distributed process the flexibility required to dynamically select new user interfaces to suit a specific user interface environment and to allow user interfaces to be directed across a distributed system, a process needs to be split into three separate but communicating physical processes [8]; a *Process Handler* (PH) to manage the functionality of the process, an *Interface Handler* (IH) to manage the user interface of the process and an *Interface Viewer* (IV) to render user interfaces and process user input data (figure 5).

When the PH forwards a new interface, the IH extracts the language type from the communication and refers to an internal look-up table to determine which IV will handle the rendering. This IV is invoked on the user's local display with the specified user interface. When the user performs an action on the user interface that causes data to be returned, the IV passes raw input data back to the IH, which converts it into a format that the PH can understand (that is, it becomes 'cooked'). The PH performs appropriate processing upon the data and decides what actions to perform in response (for example, quitting or forwarding a new interface). All data that is communicated between the IH and the PH uses the Microcosm message format [14], which allows it to be free-form and dynamically extensible.

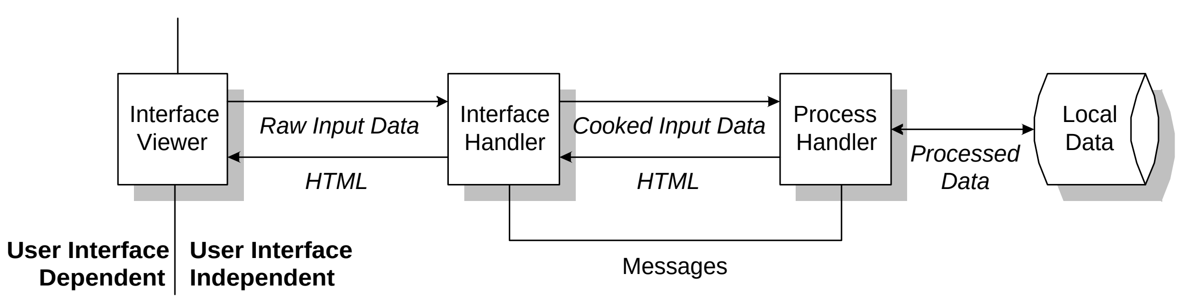


**Figure 5 : Process and Interface Handler Interactions**

To prevent the PH from becoming too involved in user interface details of a platform, a user interface environment independent display description language must be used. This will enable the PH to build and subsequently communicate new user interfaces to the IH in terms that are both descriptive but abstract. HTML [4] is such a language that is currently in wide-spread use within the WWW. HTML was created to allow user interfaces to be rendered in a GUI independent manner, and, by subsuming a core subset of the windowing metaphor (buttons, text boxes, list boxes, etc.), it provides a flexible and consistent set of display components. When the user interface is to be displayed, a HTML viewer (such as Netscape or Mosaic) can be invoked to render the interface into GUI-dependant terms. This means that only the HTML viewer needs to be GUI-dependant; both the PH and the IH can remain interface independent. The use of HTML means that processes can be developed much more rapidly, since less programming effort is required in producing the user interfaces.

Another requirement that was identified whilst developing the HCM was the idea of remotely configuring processes. Each PH may need to present an external and programmable "interface" to other processes, so that their state and, maybe, behaviour can be altered. By using the HCM, each process can be contacted and configured through this interface. By extending this concept slightly, it is very easy to see how certain control processes could perform tasks on a user's behalf, similar in concept to agents. Additionally, this configuration "interface" could be used to keep processes in a consistent and informed state. A language such as the Knowledge and Query Manipulation Language [10] would be suitable for this task.

Unfortunately, although HTML is versatile enough to be able to describe a wide range of interface types, it is not suitable for handling other media formats and display orientations, for example, three-dimensional models or the description of an interface in terms of pixels or spatial relations between interface gadgets. However, such GUI-independent protocols do exist (Virtual Reality Modelling Language [2], for example) and can easily be incorporated into the model by adding a naming tag and appropriate viewer to the IH.

# 5. Distributing Open Hypermedia

## 5.1. Encapsulating closely related hypermedia resources

In the rapidly expanding global information space a mechanism is essential for encapsulating closely related documents. The notion of a *hypermedia application* was sought to limit the scope of a resource base to a defined and manageable entity.

We define a hypermedia application (figure 6) as a binding of a suite of processes (with associated preferences and configurations), a collection of documents and an arbitrary amount of link data. Such an application can now be considered as a self-contained, publishable entity.

The author of a hypermedia application is now afforded a level of abstraction beneath which any configuration issues may be concealed. Users of published applications need not be concerned with any of these issues, but can simply augment their current environment with another logical entity. This definition means that a hypermedia application exhibits the property of reflexivity, since one application may, in turn, comprise many other applications:

application ::= {process} x {link data} x {document} x {application}

A *session* acts as a container for one or more hypermedia applications:

session ::= infrastructure x {application}

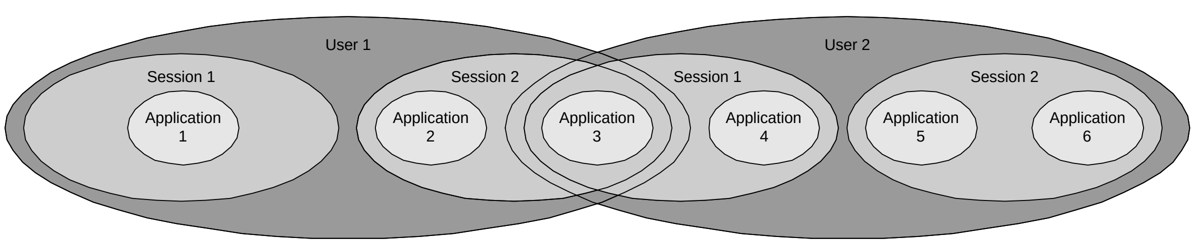


**Figure 6 : Application Encapsulation and Sharing between Users**

A user can assemble a session from arbitrary hypermedia applications. While the applications in a particular session are likely to be related, a user is free to combine any applications to produce a session that meets their current needs. For example, a commercial user might include applications relating to various different projects to compare their progress.

Each user may have a number of sessions open, each containing related applications that provide a logical context in which to work:

user ::= {session}

## 5.2. Microcosm: The Next Generation Prototype

As proof of concept, a prototype was developed to address two main areas:

- That the HCM system was suitable and appropriate for distributing a given open hypermedia system.
- That such a hypermedia system should embody the application encapsulation outlined in section 5.1.

The Microcosm model was chosen as the system to attempt to distribute, since it exhibits the majority of the requirements for open hypermedia and its modular nature lends itself well to distribution.

Within the Microcosm hypermedia model there are a number of core elements that work together to provide a minimum level of functionality:

- *External link storage*. The concept of link separation is well documented and its advantages clear when applied to open hypermedia [11,26,1]. A *linkbase* process maintains the link database and handles all link queries and creations. Multiple linkbases can exist to break document link connectivity into logical groupings.
- *Presentation and browsing*. Displaying links and allowing users to browse through links is handled by a range of *viewers*, primarily one for each class of media (text, bitmap, sound, etc.). Viewers provide the first-level interface to the user and let the user follow links and submit link queries.
- *Link selection*. The results of link queries that have been submitted by a user are handled by the *available links* process. This presents the links to the user for selection and then invokes the appropriate viewer to render the document.
- *Link authoring*. To allow links to be created within a link database, an intermediary process (the *linker*) performs pre-processing on the new link to determine its details (for example, the start and end anchor positions, the type of the link, the description of the link, etc.). Once this information has been collected, the completed link is forwarded to the relevant linkbases for creation.
- *Document organisation*. The document database (or *docuverse*) was introduced to provide a layer of abstraction between the documents themselves and the filing systems upon which they reside. By referencing each document as a unique identifier, an identifier only needs to be resolved to an actual filename when it is to be rendered (typically by a viewer).

The Microcosm TNG prototype developed using the HCM took the core concepts identified previously and modelled them as discrete, distributed processes. These processes (linkbases, viewers, available links, linkers, docuverses, etc.) were described in terms of the messages that they send and receive and the processing that they perform. Processes are considered to be peer entities due to the fact that they can communicate with other processes asynchronously. The distribution of both processes and messages is handled by the HCM subsystem. In this way, the naming of a process is kept independent to the location of the process.

Resources are distributed through the application entity to which they belong. Once a user has authored an application, they can publish it to the world at large. The Microcosm TNG system provides a mechanism for allowing remote applications to be transparently included within a user's session. Also, since an application can comprise both data and processes, this inclusion not only increases the functionality of the user's session, but also increases their available dataspace; all actions (for example, link queries) will be automatically forwarded to all of the user's local *and* remote applications.

These concepts are illustrated in figure 7. In this diagram the two users, User 1 and User 2, both have sessions running in their respective domains. In a strict local session, all link queries and link creations

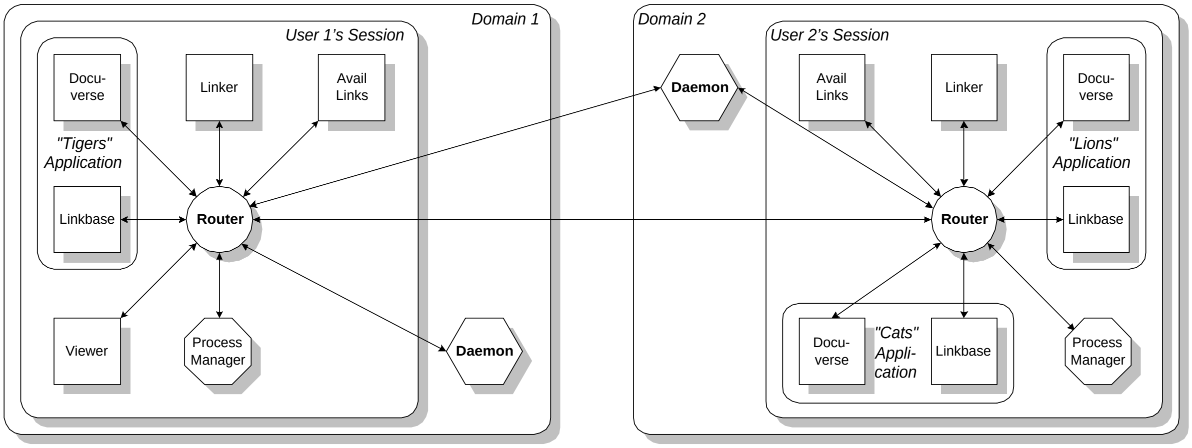


**Figure 7 : Instances of Microcosm TNG Communicating across a Widely Distributed Environment**

only apply to the applications that are bound to a particular session (for User 1, this is the "Tigers" application). User 1 and User 2 can create their applications in isolation, importing documents and media, and making links between them.

However, imagine that User 2 wishes to create a new application, called "Cats", which is composed of both the "Tigers" and "Lions" applications. This new application exists in his local session, and he makes a connection to the "Tigers" application through User 1's domain address (which he obtained previously). Once connected, User 2 has access to both applications and he can create links between them and import new documents as appropriate.

Upon completion, the "Cats" application can then be published for other people to connect to and used in a similar fashion as indicated previously. It is important to note that when users access this application, connection to the "Tigers" and "Lions" applications and hypermedia links are handled transparently.

To facilitate the heterogeneity of the user interfaces, all of the processes in the prototype were split into a Process Handler and an Interface Handler (see section 4). To provide a consistent and central interface to the processes, the Process Manager (PM) of the HCM was extended to allow each process to be configured and manipulated through it. As the PH of each process executes, a launch message is received by the PM. The initial display on the PM is a list of processes in the system, which is updated dynamically. A user can select a particular process, which instructs the PH of the selected process to display its top-level interface. From here, all data from the user interface is passed directly to the selected PH and the user can alter or interrogate the state of that process. This gives the user and other processes the ability to call forward the interfaces of both local and remote processes.

This prototype has demonstrated that distributing an open hypermedia system above the HCM shows great promise for the future. The implementation demonstrated wide-scale distribution of users, data and processing, open hypermedia functionality and integration capabilities for co-operative working environments. The next phase of development will reinforce and extend the system described here to produce an industrial-strength, distributed open hypermedia system.

# 6. Future Work

The heterogeneous user interfaces described earlier suffer from the numerous limitations of the HTML viewers. The future direction of this work lies in exploring the virtues of Java [28] for developing more flexible, heterogeneous and portable hypermedia processes. These processes would essentially be lightweight, but would take advantage of Java's object-oriented nature to provide dynamic data type support across multiple platforms.

At the time of writing a docuverse exists in a primitive form, but further work is necessary to incorporate caching, proxies and extensible support for network protocols, for example, FTP, HTTP, etc. Additionally, approaches such as Uniform Resource Identifiers (Uniform Resource Names [27] and Uniform Resource Characteristics [18]) and Harvest [5] could be integrated to aid with hypermedia application discovery on a global scale.

The HCM was designed to accommodate collaboration and as such broadcasts notification events upon users beginning and terminating sessions. This information was subsequently used to build an awareness utility, showing users within the domain. Current research examining the potential of Microcosm to support advanced CSCW features [19] will undoubtedly impact upon the future development of the Microcosm TNG system.

When disclosing resources for remote users to peruse, security becomes a genuine consideration. Authentication mechanisms are also required if users are to be charged for the time/resources they use. Preliminary work in this area has been conducted but further effort is required to develop these ideas.

Wilkins et al. [31] discuss how communities of co-operating intelligent agents can greatly assist with a variety of tasks within Microcosm. The role of agents within an open hypermedia system can be sub-divided into three categories:

- Resource location and discovery.
- Maintaining information integrity.
- Navigation assistance.

It has been recognised that the HCM and Microcosm TNG provide an ideal framework for further experimentation with agent technology and the Internet.

Technologies such as Common Object Request Broker Architecture (CORBA) [22] and Distributed Computing Environment (DCE) [23] are showing great promise for distributed application inter-operability. As part of our ongoing commitment to integrate with existing and emerging standards, these developments are among those under current investigation to evaluate their applicability to the Microcosm TNG system.

# 7. Conclusions

Distributed systems have an inherent reliance upon an underlying communication substrate, and it is crucial to overall performance that this component is reliable, efficient and scaleable. It is clear that the benefits afforded through the flexibility of these systems owe much to the relative strengths of the communication layer.

A modular design based on co-operating processes can provide the extensible functionality afforded by such distributed systems. The self-contained nature exhibited by these processes predisposes them to being distributed over a heterogeneous network of machines thus promoting concurrent processing, and hence the promise of achieving greater efficiency.

The design of the HCM has embraced many of the requirements of a distributed communication layer and provides an open framework for the rapid development of powerful distributed systems.

The client/server model, the predominant architecture among systems using the Internet, can restrict the development of truly distributed systems. This paradigm makes it very difficult for current information services to take the initiative in delivering fresh information to the user. Furthermore, some processes need to act as both client and server depending upon whom they are interacting with. The open model of the HCM allows systems to benefit from a more flexible peer-to-peer based alternative.

The rapid development of the HCM-based hypermedia system described in this paper illustrates the ease with which the HCM allows distributed systems to be built, and as such enables Microcosm TNG to provide a degree of flexibility not found in other distributed hypermedia systems. As one would expect, collaboration between users is promoted together with strong support for the sharing of information. In particular, the ability to encapsulate a group of distributed processes with their respective resources and publish them as a single entity provides a welcome degree of abstraction. The hypermedia application conceals configuration details and provides a scaleable mechanism for building and composing new applications.

In addition, the heterogeneous nature of the hypermedia system promotes efficient use of available resources. Processes may easily be allocated to machines with the most suitable facilities, or which are being under-utilised at any specific time.

# References


[1] ANDREWS, K., KAPPE, F. and MAURER, H., Serving Information to the Web with Hyper-G. *In:* Computer Networks and ISDN Systems, Volume 27, Number 6, pages 919-926, 1995.

[2] BELL, G., PARISI, A. and PESCE, M., Virtual Reality Modelling Language Specification 1.0.

[3] BERNERS-LEE, T., CAILIAU, R., GROFF, J. and POLLERMANN, B., World-Wide Web: The Information Universe. *In*: Electronic Networking: Research, Applications and Policy, Vol. 2 No 1, Spring, Meckler Publishing, Westport, CT, USA, pages 52-58, 1992.

[4] BERNERS-LEE, T. and CONOLLY, D., Hypertext Mark-up Language.

[5] BOWMAN, C., MANBER, U., DANZIG, P. B., SCHWARTZ, M. F., HARDY, D. R. and WESSELS, D. P., Harvest: A Scaleable, Customisable, Discovery and Access System, CU-CS 732-94, Department of Computer Science, University of Colorado, 1995.

[6] BURGER, A. M., MEYER, B. D., JUNG, C. P. and LONG, K. B., The Virtual Notebook System. *In*: Hypertext 91, Proceedings of Third ACM Conference on Hypertext, San Antonio, Texas, USA (December), ACM Press, pages 395-402, 1991.

[7] DALE, J., The Communication Routines - A Network Layer Communication Model, Multimedia Technical Report M95/2, Department of Electronics and Computer Science, University of Southampton, 1995.

[8] DALE, J., Distributed User Interfaces: Achieving User Interface Heterogeneity in a Distributed Environment, Multimedia Technical Report M96/1, Department of Electronics and Computer Science, University of Southampton, 1996.

[9] DAVIS, H. C., HALL, W., HEATH, I., HILL, G. and WILKINS, R. J., 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), ACM Press, pages 181-190, 1992.

[10] FININ, T., et al, Knowledge Query and Manipulation Language Specification.

[11] FOUNTAIN, A., HALL, W., HEATH, I. and DAVIS, H. C., Microcosm: An Open Model with Dynamic Linking. *In*: A. Rizk, N. Stroetz and J. André, *Eds*. Hypertext: Concepts, Systems and Applications, Proceedings of the European Conference on Hypertext, INRIA, France (November), pages 298-311, 1990.

[12] GOOSE, S., Distributed Open Hypermedia Systems, PhD Mini-thesis, Department of Electronics and Computer Science, University of Southampton, 1995.

[13] 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), ACM Press, pages 191-200, 1992.

[14] HEATH, I., An Open Model for Hypermedia: Abstracting Links from Documents, PhD Thesis, Department of Electronics and Computer Science, University of Southampton, 1992.

[15] HILL, G., WILKINS, R. J. and HALL, W., Open and Reconfigurable Hypermedia Systems: A Filter Base Model. *In:* Hypermedia, 5(2), pages 103-118, 1993.

[16] HILL, G. and HALL, W., Extending the Microcosm Model to a Distributed Environment. *In*: ECHT '94 Proceedings, Edinburgh, Scotland (September), ACM Press, pages 32-40, 1994.

[17] MALCOM, 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, Texas, ACM Press, pages 13-25, 1991.

[18] MEALLING, M., Specification of Uniform Resource Characteristics, April, 1994.

[19] MELLY, M., and HALL, W., Co-operative Work in Microcosm, Computer Science Technical Report, Department of Electronics and Computer Science, University of Southampton, 1995.

[20] NELSON, T., Literary Machines 87.1, published by the author, Mindful Press, 1987.

[21] NYE, A., Xlib Programming Manual Volume 1 (3rd edition), O'Reilly & Associates Inc., 1993.

[22] Object Management Group, The Common Object Request Broker: Architecture and Specification, OMG Technical Document Number 91-12-1, Revision 1.1, December 1991.

[23] Open Software Foundation, Distributed Computing Environment Overview, 1992.

[24] PEARL, A., Sun's Link Service: A Protocol for Open Linking. *In*: Hypertext '89 Proceedings, Pittsburgh, USA (November), pages 137-146, 1989.

[25] 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, USA (November), pages 129-136, 1989.

[26] SMITH, K. E. and ZDONIK, S. B., Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems. *In:* OOPSLA '87 Proceedings (October), pages 452-465, 1987.

[27] SOLLINS, K. and MASINTER, L., Requirements of Uniform Resource Names, March, 1994.

[28] Sun Microsystems, Java Programming Reference Manual, Sun Microsystems Inc.

[29] 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), ACM Press, pages 251-261, 1992.

[30] WILKINS, R. J., HEATH, I., and HALL, W., A Direct Communication Model for Process Management in an Open Hypermedia System, Computer Science Technical Report 93-14, Department of Electronics and Computer Science, University of Southampton, UK, 1993.

[31] WILKINS, R. J., DeROURE, D. C., HALL, W. and DAVIS, H. C., The Role of Agents in Multimedia Information Systems. *In* Proceedings of the Intelligent Agents and the Next Information Revolution, Manchester, UK (May), pages 14-23, 1995.