Concepts and Architecture for Loosely Coupled Integration of Hyperbases *

Teresa Chambel[†] **Carmo Moreno**[‡] **Nuno Guimaraes**[‡]

Pedro Antunes[‡]

INESC Instituto de Engenharia de Sistemas e
Computadores, R. Alves Redol, 9 - 6^{o} - 1000 Lisboa - Portugal, Tel.+351-1-3100000.
Direct Line: 3100223, Fax: +351 1 525843.
e-mail:...{mtc,mcm,nmg,paa}@inesc.pt

From the computational viewpoint, organizations, and specially medium to large scale organizations, are decentralized structures, maintaining heterogeneous information, manipulated by all kinds of applications. Organizational partition, security constraints or performance requirements induce the creation of related but independent information repositories.

Hypertext and hypermedia systems provide a flexible way of structuring large collections of heterogeneous information items. A fundamental characteristic of hypermedia systems is navigation, acting as an augmenting factor. In the organizational environment, hypermedia acts as a glue that binds together the information repositories. The structuring and augmenting roles of hypermedia provide a higher degree of homogeneity in the access to the organizational information system and allow the establishment of relations that would otherwise be left unperceived.

Hypermedia has to be introduced in a loosely coupled fashion, preserving the autonomy of individual systems, and avoiding the creation of entangled universal webs. This paper describes an architecture designed to address this issue, emphasizing the role of query based mechanisms and group communication, as an alternative to static linking between independent hyperbases.

1 Introduction

From the computational viewpoint, organizations, and specially medium to large scale organizations, are decentralized structures, maintaining heterogeneous information, manipulated by all kinds of applications. Organizational partition, security constraints or performance requirements induce the creation of related but independent information repositories.

One of the most relevant facets of hypertext and hypermedia systems is the flexible structuring power of large collections of heterogeneous information items. Hypermedia documents, designed for educational purposes, are nowadays clear examples of the structuring function applied to diversified chunks of multimedia information. These information chunks become intelligible through the superimposition of an appropriate structure, and acquire a specific meaning when presented as a whole (*gestalt*). In this context, the fun-

* Based on a paper presented at ACM Hypertext'93 Workshop on Hyperbase Systems. This work was partially supported by the CEC, through Esprit BR 6360 (Broadcast).

† Science Faculty of the University of Lisboa.

‡ Technical University of Lisboa.

damental characteristic to be stressed is navigation. The navigational nature of hypermedia systems acts as an augmenting factor, as foreseen by pioneering proposals by Bush [6] and Engelbart [14].

The definition, design and implementation of hypermedia systems has therefore been focused on the augmentation of the human capabilities in terms of information creation, storage and manipulation. On the other hand, the use of hypermedia systems by groups of users emerged as a natural evolution and led to a leading role of hypermedia in the domain of computer supported collaborative work.

The impact of hypermedia in the work of groups has to be evaluated in the context of the organizational structures and dynamics. Organizations adopt different configurations and flows, depending on their goals, evolution and adaptation to the external environment [18].

In the organizational environment, hypermedia has the potential to act as a glue that binds together the information repositories. The structuring and augmenting roles of hypermedia provide a higher degree of homogeneity in the access to the organizational information system (seen as the aggregation of all the elementary information repositories) and allow the establishment of relations that would otherwise be left unperceived.

The effective use of hypermedia as an *organizational information glue* imposes the definition of an open architecture. The openness of the hypermedia architecture derives from both the application design (as sustained in [24,16]) and from the mechanisms for interconnection of distributed hypermedia structures [11,30]. To address this later issue we assume the following requirements:

- The information repositories are located in a distributed environment.
- The information contained in the repositories is "untouchable". It is based on specific schemas, dependent from application specific requirements.
- The number of repositories grows as the organization becomes more complex.
- Autonomy is paramount, both for organizational flexibility and technical viability (upgrading, maintenance).

The assumption of these requirements is strengthened by a general perspective of a growing ubiquity of computing systems, provided by mobile systems and consumer computing. The conclusion is that the use of hypermedia has to be introduced in a loosely coupled fashion, preserving the autonomy of individual systems, and avoiding the creation of entangled universal webs.

This paper describes an architecture designed to address this issue. The architectural guidelines are:

- The information repositories are elementary units, encapsulated/structured by an individual hyperbase. This provides a common access model.
- The integration of information repositories is achieved through interactive cooperation of the individual hyperbases.
- The mutual knowledge between hyperbases and the relations that are established during global navigation are essentially transient. They are created and used during a given period of time (*a session*) and forgotten afterwards.
- The underlying infrastructure is based on *group*-oriented communications [3,4,28,29].

The next section of the paper reviews some related work, including current approaches to distributed hypermedia. The following section illustrates how we achieve loose coupling between hyperbases, enumerates some solutions and identifies open issues. The following section describes the architecture of the *HyperGroup* system. It also refers to implementation and application current prototype. Last section unveils new problems to be solved and draws some

conclusions.

2 Related Work

2.1 Distributed Databases and Multidatabases

Distributed Database Systems, and in particular, those concerned with preserving autonomy of the databases (Federated Databases and Multidatabases [21] [5]) share some of the motivations and problems with the approaches of hyperbase integration. While Distributed Databases are typically designed in a top-down fashion as a distributed tightly coupled system, multidatabases are used to integrate existing databases in a distributed environment. Common problems include: dealing with heterogeneity, schema inconsistency, concurrency control, global query processing and helping users in finding the information they need.

Distributed databases usually deal with data model and repository heterogeneity through the adoption of gateways that perform translations between different systems [15,17].

2.2 Distributed Hypermedia Systems

Distribution in hypermedia systems has evolved due to both architectural decisions and design approaches. Single user systems became distributed through enhancements of the architecture. Distributed hypertext has been a design goal in other systems. Moreover, the development of the networking capabilities available to the community of users is suggesting a much wider use of hypermedia-based approaches in the distributed environment. In this section we review this evolution and stress the issues we believe to be more relevant for the work presented in this paper.

Single Server solutions The architectural developments of early hypertext and hypermedia systems led to the support for simple distribution mechanisms. As an example, the KMS approach [1], relies on the use of a distributed file system, namely NFS, and becomes therefore distributed. The use of storage backends as in Neptune [12,7], known nowadays as *hyperbases* [26,30], separates the storage of the hypermedia structure from the presentation. The immediate step is to implement the storage and presentation as a client-server architecture, thus introducing a simple notion of distribution.

Multiple Server approaches Multiple server systems allow nodes to reside in different machines and the establishment of links between them. Some of these, such as: Sun's Link Service [23], Distributed DIF and Virtual Notebook System [27], store links and node location in a database. Plane Text [8] keep links and nodes in Unix files. Other systems, like Telesophy [25] and those proposed by Xanadu [19] and Open Hyperdocument Systems [13], share the goal of linking the knowledge base of a large scale organization, a community or a nation.

The approach presented by Noll and Scacchi [20] claims to be radically different from previous ones in trying to provide transparent access to heterogeneous information repositories while maintaining their autonomy. The Distributed Hypertext (DHT) architecture is based on a client-server model and includes four components:

- Common Hypertext Data Model - to the clients, a common model describes all the objects in the information space: a set of basic objects

(nodes, links, attributes of links or nodes) and a set of operations (create, update, delete an object or traverse a link).

- Communication Protocol - implements the operations defined by the data model and provides the mechanism for moving objects between clients and servers.
- Servers - include two components: a Gateway process that transforms hypertext operations into local access operations and local information objects into hypertext nodes and links; and an Information Repository that contains the information to be accessed (a file system, a database or a special purpose storage manager). The pre-existing applications continue to function as before, and gateways are presented to the information repositories as regular local applications that access its data.
- Client Applications - applications and specific styles of user interfaces to the primitive operations provided by the hypertext data model.

To address this kind of integration problem, two broad approaches have been used: distributed file systems and heterogeneous databases. File systems offer the flexibility and ease of use but they constrain organization to hierarchies of directories. Databases offer many high-level features, including powerful organization primitives and multiple views, but sacrifice autonomy for transactions.

As pointed out by their authors, the major advantages in using an hypertext model are its inherent transparency, organization and flexibility, making it easier to deal with the integration of heterogeneous information repositories, while preserving repository autonomy.

Hypertext with its user interaction facet provides transparency, regardless of the data types in the information space; with its data representation facet provides a powerful and extremely flexible way of organizing information; with its data storage facet, provides the key to build an integrated hypertext in an autonomous and heterogeneous environment. There is no need for a global transaction manager that might infringe on local control and authority, since any transaction is restricted to read or update operations on a single node or link and to create operations on a specific server. A hypertext storage mechanism can manage diverse data types and be implemented on a variety of storage managers.

The major disadvantage of the proposed approach is the need to implement a new gateway for each new repository. This is however a common limitation in other integration strategies.

2.3 Information Discovery Services

The Internet has the potential of being an important researcher's information source. However, its growing size and complexity are considerable obstacles and a paradigmatic navigation challenge. To overcome this problem and assist users in finding relevant information, a number of *resource discovery tools* have recently been developed [10]. These tools provide users with browsing, searching and organizing facilities. Users can navigate through the available information space, or perform query-based searches to locate relevant data.

Resource discovery tools differ in the facilities provided (browsing, locating, querying), in the way they organize the information (as a generalized directed graph, using indices), in the granularity of information (files, documents, info about users, file names, individual messages in a mail file), in the place data is stored (distributed among geographically dispersed servers, centralized and replicated indexing databases, distributed indexing databases). Some of them implement directory services and let users query for relevant servers.

Among them, WWW - World Wide Web [2] and Indie [9] are hypertext systems. In WWW users have to follow pre-defined links, but can choose a place to start from. Creation is allowed for links departing from nodes owned by the user. In Indie, though, because it is built on top of the Distributed Hypertext (DHT) scheme described above [20], users can organize their information space into a distributed hypertext. In particular, they can add links to help them find the information in the future without having to repeat the entire discovery process.

The initial development of these Internet discovery of services was done independently from each other, but they are evolving in the direction of interoperability, where users of one discovery service can access information available through the others. Some of them already achieve this to a certain extent.

3 Concepts and Mechanisms for Loosely Coupled Integration

As stated above, our goal is to allow navigation on a universe of loosely coupled and heterogeneous hypermedia bases, avoiding explicit knowledge of the available servers, and avoiding the establishment of static links or relations that hinder mobility and local autonomy.

To address this goals we defined the *HyperGroup* model. It is a client-server model based on both group communication and hypermedia paradigms.

Typical approaches to the construction of multiple server distributed systems, such as our environment, are based on point to point communication, complemented with a definition of a name space, usually built upon logical names, subsequently translated to server addresses by a special component of the system, the name server.

Even if the indirection provided by the name server allows some degree of flexibility, this solution implies agreements on "well-known" addresses, or on pre-defined logical names. We believe these implications to be contrary to our goals. Our solution was to use a support for group communication. According to this approach, the local hyperbases just have to know the group of peer servers they want to belong to and connect to it. Group membership and group communication does not require explicit mutual knowledge. It just provides a "sense of belonging" and a "communication space" with whoever shares that space.

Hyperbase servers become members of a server group - *HyperGroup*. In this context, *HyperGroup* model guidelines are:

- Servers willing to make their service available join the group. They don't have explicit knowledge of other hyperbase servers - keeping their autonomy.
- Clients get information as group clients. They don't have to know which hyperbases belong to the group.
- Clients can operate on the group (ask for group membership, select hyperbase subgroups, etc.) or on hyperbases (queries, navigation, etc.).
- The system is dynamic in the sense that it allows the servers to "plug" in and out the system at any time. The clients are aware of these changes - they do group monitoring.

Figure 1 exemplifies the communication environment underlying the *HyperGroup* model.

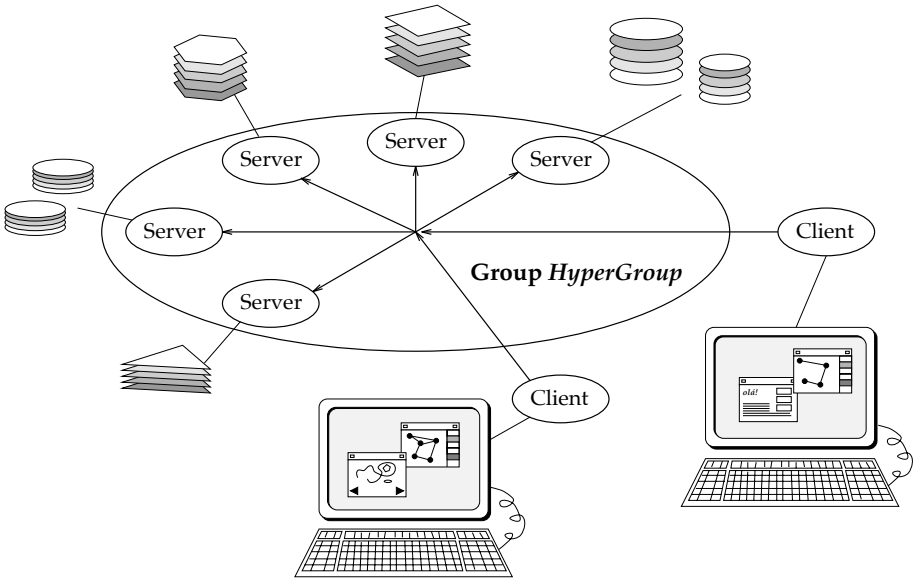

Figure 1. Overall Picture of the Navigation Environment

3.1 Local Hyperbases to handle Hyperbase Heterogeneity

The distributed hypermedia environment presented in this paper is composed of discrete and independent hyperbases. These hyperbases are called *SOHO*[1] bases. The relevant characteristics of *SOHO* are:

- *SOHO* bases define a simple hypermedia model, based on the *HAM* model [7]. This includes object classes like *graph*, *context*, *node*, and *link*. All the objects belonging to these classes may have *attributes*. The *Composite* class is being introduced but we will consider this evolution orthogonal to the work being described here. In the scope of a *graph*, each object has a unique identifier (OID).
- *SOHO* provides query facilities. The query facilities are both attribute-based and content-based. This means that, for example, nodes can be queried for based on particular attributes, or based on content characteristics. This later case only applies to text nodes since no search facilities on other media are available.
- Navigation in *SOHO* is supported by link traversal, history and path mechanisms. Also, the integration of query and search techniques enables the construction of dynamic paths. Paths are essentially persistent collections of object (typically node) identifiers. These may be created explicitly by a user, or implicitly through the query mechanisms.
- *SOHO* is implemented on three different supports. The simplest and most immediate is the U*ix file system, where nodes are directories, links are U*ix symbolic links and attributes are stored in files. Although very inefficient, this implementation allows easy integration of facilities like some degree of distribution (by using NFS, for example), protection and ownership attributes at object level (derived from U*ix mechanisms), and object (file) locking.

  The lightest implementation is based on indexed files, provided by the *sdbm* library. In this case, the wins are a greater portability and faster access to the information, as a counterpart to a loss of the protection and locking mechanisms that would have to be reimplemented at the record

[1] Storage Of Hypermedia Objects

level.

The most functional implementation is based on the ONTOS object oriented database. The fundamental difference of this last implementation is the availability of the SQL language, provided by the ONTOS system. This query language is a sound basis for the implementation of the query facilities defined by the *SOHO* interface.

As described, the *SOHO* bases are in fact an interface to some kind of storage system, providing uniform access to the information elements, based on a hypermedia model. A comparison with the systems mentioned above allows us to see these *SOHO* bases as similar to the DHT *Gateways*. As mentioned before, distributed databases also adopt gateways to deal with this kind of heterogeneity.

3.2 New Requirements for Loosely Coupled Integration of Hyperbases

Some new requirements will lead to the extension of the hypermedia model used at hyperbase level, towards the *HyperGroup* model.

3.2.1 Group Management

Server space is, in *HyperGroup*, a group of servers. It should be offered group management facilities, to deal with it. These facilities are mainly concerned with membership management: group membership information, server information (e.g. site), subgroup creation, group monitoring (members joining and leaving the group). A member may leave intentionally or due to a failure.

3.2.2 Group Selection

The server space is formed by a set of hyperbases belonging to a group. Our goal is to keep the basic group scalable, i.e. able to support a large number of hyperbases accessing the organization's information system. This illustrates an amorphous set of servers. Filtering or selection mechanisms are then required to structure this mass. Assuming, as we do, that this selection is directed by the user wishing to navigate in the hypermedia space, we considered two kinds of mechanisms to implement group selection:

- Explicit selection. Hyperbases have names. If these names are meaningful to the user, a subgroup can be formed by explicit selection of a set of servers. The knowledge about the usefulness of a server is the user's responsibility. It can be derived from a good guess, empirically classified names or, most commonly, from previous experience.
- Implicit selection. A generic query can be multicasted to the group, in order to create a subgroup with those hyperbases that possess answers to the query. In this context, a particularly interesting query is based on the following: Hyperbases have attributes and can be queried. The hyperbase attributes are essentially attributes associated to the graphs managed by a *SOHO* base. These queries based on some graph attributes are meant to ask the universe of servers which ones "think" will be able to respond to particular kinds of queries, when these come to be performed. We call these mechanism "Query on queries".
- Help systems or Catalogs play an important role in the *bootstrap* of the navigation process. These Help systems contain summaries or descriptions provided by the hyperbases structure and information, that allow users to select them, either implicitly or explicitly. In an organizational setting, the vocabulary is generally consistent among users and therefore the terminology used in these catalogs is expected to have low levels of ambiguity.

3.2.3 Object Access

Once a subgroup has been selected, either implicitly or explicitly, queries can be performed on objects. At this stage, group queries are essentially performed as local queries with the difference that they are multicasted to the subgroup. The result of a query is a set of object identifiers, for example nodes which textual content includes a given string (if a content-based query has been issued).

The object identifiers that are returned as a result of a multicasted query are, of course, not unique. Identifiers are designed to be unique in the scope of a local hyperbase. When object identifiers are gathered in this way, they have to be transformed for further manipulation. Broadly speaking, this is achieved through "concatenation" of the group member identification with the local identification. One must be aware that these newly formed identifiers are essentially transient and meant to allow operations like visualization or editing, in the time frame of a navigation session.

At this point, we have a set of object identifiers, references to nodes and links in multiple servers. Navigation maps can be produced with this information. The most likely operations are node selection and visualization, and link display and traversal.

3.2.4 Query-based Navigation Aids

A fundamental problem in hypermedia systems is the design of mechanisms to assist navigation. History, paths or guided tours have been proposed as improvements in this direction. In our context and goals, these mechanisms have the handicap of relying on the object identification mechanism. Essentially, they are all (persistent) records of object identifiers, or names, that a user has visited or is expected to visit.

Query History Our approach is to avoid a global space and to rely on query-based operations to obtain object identifiers. Taking this approach further, we considered that a persistent record of a set of queries would be adequate to perform the navigation assistant role. This solution was also adopted due to the belief that, in a dynamic and loosely coupled environment, the objective of keeping a history record or providing for a guided tour, was not so much to define a fixed set of particular objects, but instead to define a fixed semantics of the navigation operations. This navigation semantics (materialized in a fixed set of queries) results in different sets of objects at different times and different places. Using this approach, a guided tour becomes essentially a "query replay" mechanism, that has some undeterministic results when the answers are observed rigorously in terms of object identifiers.

3.2.5 Hints or Translation Mechanisms

One of the problems that we noticed to be likely to occur in the query-based mechanisms we have been describing is the inconsistency and conflict between the "key values" in the different hyperbases, the names of the attributes which are queried for. The practical result of the inconsistencies and conflicts is the difficulty of knowing which attributes have related meanings in multiple hyperbases, and also whether two attributes with the same name have the same meaning. In general, this is a consequence of blending together heterogeneous schemas [21]. This approach is also similar to suggested schema integration mechanisms for multidatabases [5].

Our preliminary solution to this problem is to allow each hyperbase to maintain *hints* to other hyperbases. These hints provide orientation to further

queries. A hint, placed on a given hyperbase, could be expressed either explicitly or implicitly. The first case translates attribute names directly between hyperbases. The second mechanism translates attribute names based on conditions that are satisfied by other hyperbases. The hyperbases that satisfy the conditions are found, once again, through queries.

The first method is a concession to the introduction of static information (the names of the hyperbases). The second method maintains the virtualness of the relation between the hyperbases. Both cases however, illustrate the possibility of building up some adaptive behavior, allowing the creation of hints based on previous usage.

Virtual Links A complementary mechanism can also be used at *node* level, to implement *virtual links*. In that case, a node keeps information to trigger subsequent queries or searches in the universe composed by the several hyperbases (similar to the searches in common hypertext systems). In the simplest case, this information is composed by an attribute-value pair that is used to build the query.

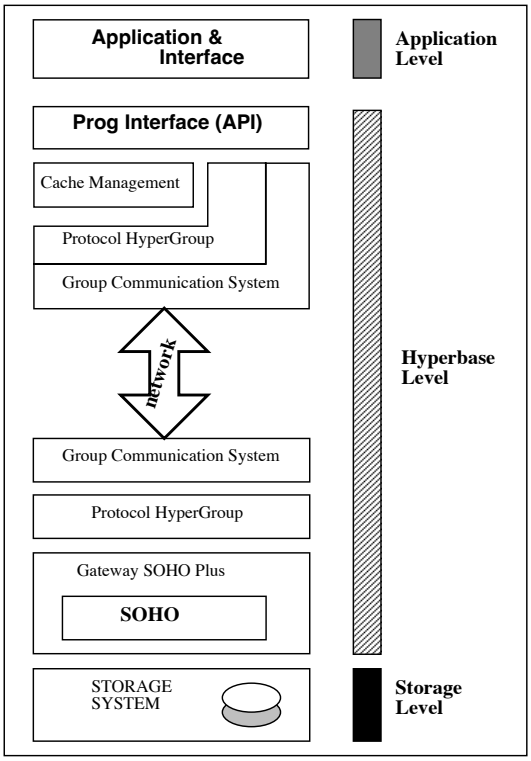

Figure 2. Generic Architectural Components

4 Architecture

The *HyperGroup* system materializes the ideas and concepts underlying the *HyperGroup* model. It was designed according to a client-server architecture based on a group communication system. **Clients** include applications and user interfaces for *HyperGroup* model. Contrary to local hyperbases, where content information is, not necessarily but typically, stored near the local hyperbase, in

our case, information has to be brought to the user. This may cause transmission of large amounts of information and therefore, large delays and network overhead. Clients do hyper object caching for efficiency purposes. **Servers** do information repositories management and include *gateways* to translate model operations to local access operations and local information to objects from the model, allowing the integration of heterogeneous information repositories. Clients and servers communicate through **HyperGroup Protocol**, which supports the model operations and provides for object exchange between client and server. Figure 2 shows the architectural components of *HyperGroup*.

Figure 3 shows the implementation diagram. *HyperGroup* is implemented in C++, using OSF Motif and EdGar (a Graph Editor developed at Inesc [22]) for the user interface, SOHO for information repository gateway (section 3.1), and ISIS [3,4] as the group communication support system. ISIS is a toolkit for distributed programming built around process groups and reliable group multicast. ISIS supports several styles of process group, and a collection of group communication protocols spanning a range of atomicity and ordering properties.

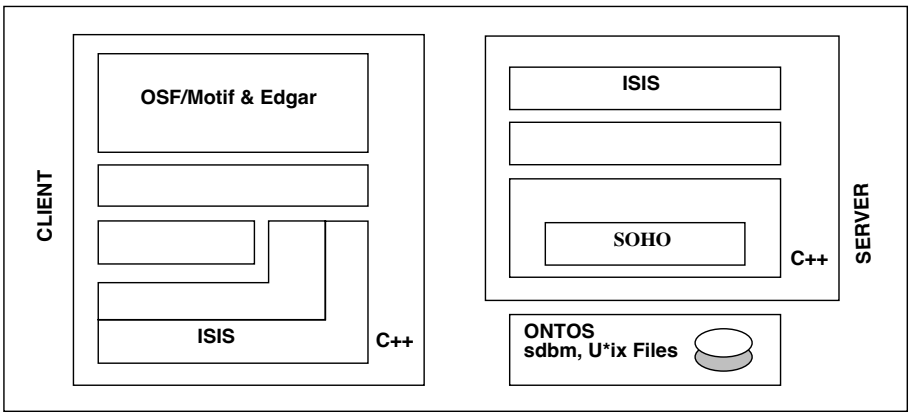

Figure 3. Implementation Architecture

Figure 4 presents a list view for an hyperbase group, a query nodes by attribute dialog and a graph hyperbase view.

5 Conclusions and Future Work

The architecture that was presented in this paper allowed us to conclude that loosely coupled mechanisms can be defined and implemented to provide access to autonomous information repositories. Some of the more detailed conclusions that can be drawn from the design and preliminary evaluations are:

- The group facilities provided by the communication infrastructure are an adequate programming paradigm for handling the distribution aspects. Issues like fault tolerance (supported to some extent by ISIS) were left out of the paper but they have a significant impact on design, implementation and run-time behavior of the system.
- The *link* mechanism can be replaced by a set of *query-based* mechanisms when used across hyperbases. The adequacy of this replacement depends on the degree of determinism that is required in multiple navigational operations, and this should be considered the fundamental *trade-off*.

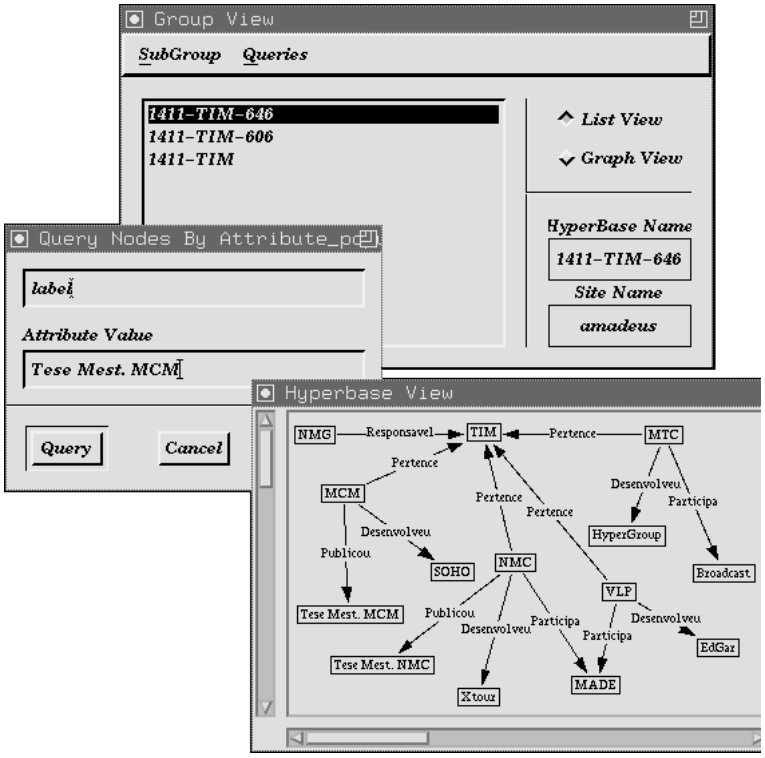

Figure 4. Group and Hyperbase Views

Strong requirements for deterministic navigation[2] can only be satisfied through static information (*links*). This is the approach proposed in [20]. Although we believe that the maintenance of this static information is required in some cases, we also realize that it turns the universe into a global information space, with intricate mutual dependencies, becoming harder and harder to manage.

- The navigation support mechanisms that are based on the primitive *queries* are designed with the goal of being "semantically consistent". This means that a *history-replay*, seen as a *path*, has to lead to similar results as the original navigation. During the registration of the *path*, local navigation (performed through local and static link traversal) has to be filtered out or registered locally. Further experience is required to produce an adequate solution to this problem.

The *query-based* approaches that were described along the paper open the way to new and relevant issues. Some of them are:

- Contextual queries. We believe this issue to be a counterpart of contextual navigation and contextual searches, and the integration of techniques that originate in information retrieval systems should be considered in the future.
- Inferencing mechanisms for provision of adequate hints are another important direction.

We intend to assess the proposed approach in a large scale environment, considering large amounts of information, objects, hyperbases, servers, and users in either local and wide area networks.

[2] *WYGNIWYGL - What You Get Now Is What You Get Later* if a new paradigm has to be found.

A key issue to be addressed is : How large can groups become in order to keep the informations space manageable ?

This issue should be addressed in three aspects :

- Communication issues – Group communication protocols, buffering, etc. restrain the group dimension.
- Information manageability by users – Selection of relevant servers and relevant information may be highly dependent on the group size.
- User interface – On the first hand, the traditional problem of orientation is exacerbated by the consideration of large and unstable spaces. On the other hand, the representation of links or hints has to be thoroughly evaluated and broadly experimented.

References

[1] R.M. Akscyn, McCracken D.L., and Yoder E.A. KMS : A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of ACM*, 31(7):820–835, July 1988.

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

[3] K. Birman, R. Cooper, T. Joseph, K. Marzullo, M. Makpangou, K. Kane, F. Schmuck, and M. Wood. The ISIS System Manual, Version 2.0. Technical report, The ISIS Project, Dept. of Computer Science, Cornell University, Ithaca, September 1990.

[4] Kenneth P. Birman, Robert Cooper, and Barry Gleeson. Programming with process groups: Group and multicast semantics. Technical Report TR-91-1185, Cornell University, Ithaca, USA, January 1991.

[5] M.W. Bright, A.R. Hurson, and S.H. Pakzad. A Taxonomy and Current Issues in Multidatabase Systems. *Computer*, March 1992.

[6] V. Bush. As We May Think. In Irene Greif, editor, *Computer Supported Collaborative Work : A Book of Readings*, pages 17–35. Morgan Kaufman Publishers, 1988.

[7] B. Campbell and J. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Communications of ACM*, 31(7):856–861, July 1988.

[8] J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, pages 17–41, September 1987.

[9] P. Danzig, S. Li, and K. Obraczka. Distributyed Indexing of Autonomous Internet Services. Technical report, University of Southern California, 1992. Available by anonymous ftp from jerico.isc.edu:pub/Indie/jcs.ps.Z.

[10] P. Danzig, K. Obraczka, and S. Li. Internet Resource Discovery Services. Technical report, University of Southern California, 1992. Available by anonymous ftp from jerico.isc.edu.

[11] H. Davis, W. Hall, I. Heath, G. Hill, and R. Wilkins. Towards an Integrated Information Environment with Open Hypermedia Systems. In M. Nanard D. Lucarella, J.Nanard and P.Paolini, editors, *Proceedings of Echt'92, 4th ACM Conference on Hypertext*, pages 181–190, Milano, Italy, November 1992. ACM.

[12] N. Delisle and M. Schwartz. Neptune : a Hypertext System for CAD Applications. Technical Report CR-85-50, Computer Research Laboratory, Tektronix, 1986.

[13] D. Engelbart. Knowledge-Domain Interoperability and an Open Hyperdocument System. In *Proceedings of Computer Supported Collaborative Work. ACM.*, pages 143–156, 1990.

[14] D. Engelbart and W. English. A Research Center for Augmenting Human Intellect. In Irene Greif, editor, *Computer Supported Collaborative Work : A Book of Readings*, pages 81–103. Morgan Kaufman Publishers, 1988.

[15] Rob Goldring. The Long Road to Heterogeneous Distributed DBMS. *Database Programming & Design*, July 1990.

[16] N. Guimaraes. HOT: a Generic Hypermedia Toolkit. In *Proceedings of the TOOLS'90 Conference, Paris*, June 1990.

[17] Michael Krasowski. Why Choose Distributed Database ? *Database Programming & Design*, March 1991.
[18] H. Mintzberg. *The Structuring of Organizations*. Prentice Hall, 1979.
[19] T. Nelson. Literary Machines. Mindfull Press, 1990.
[20] J. Noll and W. Scacchi. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. *IEEE Computer*, December 1991.
[21] M. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1991.
[22] Vasco Lopes Paulo. EdGar - Ferramenta Interactiva de Criação e Manipulação de Grafos. Instituto Superior Técnico, Portugal, November 1991.
[23] A. Pearl. Sun's Link Service: A Protocol for Open Linking. In *Proceedings of Hypertext'89, ACM*, pages 137–146, November 1989.
[24] J.J. Puttress and N. Guimaraes. The Toolkit Approach to Hypermedia. In A. Rizk, N. Streitz, and J. Andre, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of ECHT'90, Paris*, pages 25–37. Cambridge University Press, November 1990.
[25] B. R. Schatz. Telesophy: A System for Manipulating the Knowledge of a Community. In *Proceedings of Globecom 87. ACM New York*, pages 1181–1186, 1987.
[26] H.A. Schuett and N. Streitz. HyperBase: A Hypermedia Engine based on a Relational Database Management System. In A. Rizk, N. Streitz, and J. Andre, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of ECHT'90, Paris*, pages 95–107. Cambridge University Press, November 1990.
[27] F. M. ShipmanIII. Distributed Hypertext for Collaborative Research: The Virtual Notebook System. In *Proceedings of Hypertext'89. ACM*, pages 29–35, November 1989.
[28] P. Veríssimo and L. Rodrigues. Group orientation: a paradigm for distributed systems of the nineties. In *Proceedings of the 3rd Workshop on Future Trends of Distributed Computing Systems*, Taipe, Taiwan, April 1992. INESC AR/67-92.
[29] Werner Vogels, Luís Rodrigues, and Paulo Veríssimo. Fast group communication for standard workstations. In *Proceedings of the OpenForum'92 Technical Conference*, Utrecht, the Netherlands, November 1992. EurOpen, UniForum.
[30] U.K. Wiil and J.J. Leggett. Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. In M. Nanard D. Lucarella, J.Nanard and P.Paolini, editors, *Proceedings of Echt'92, 4th ACM Conference on Hypertext*, pages 251–261, Milano, Italy, November 1992. ACM.