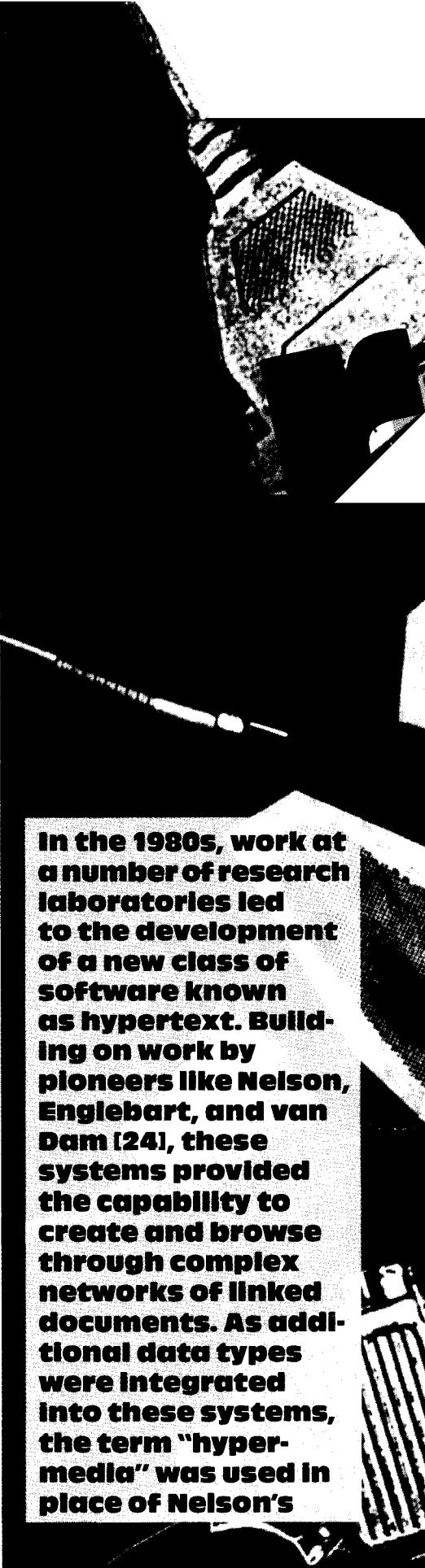


IRIS

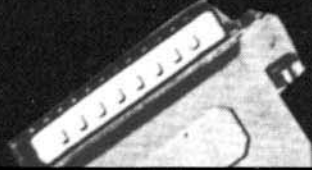
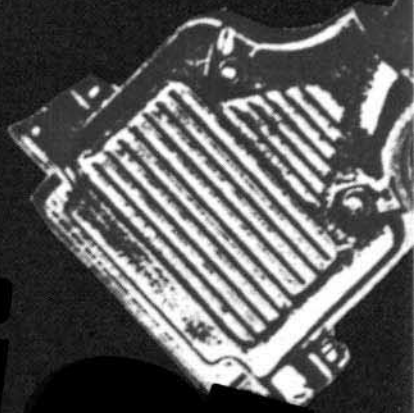
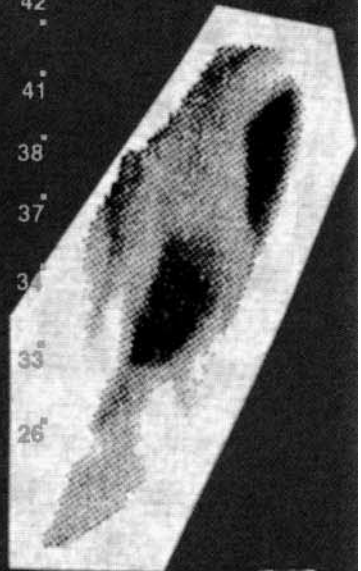
HYPER

Bernard J. Haan
Paul Kahn
Victor A. Riley
James H. Coombs
Norman K. Meyrowitz



In the 1980s, work at a number of research laboratories led to the development of a new class of software known as hypertext. Building on work by pioneers like Nelson, Englebart, and van Dam [24], these systems provided the capability to create and browse through complex networks of linked documents. As additional data types were integrated into these systems, the term "hypermedia" was used in place of Nelson's

medicoidea services



IRIS Hypermedia Services

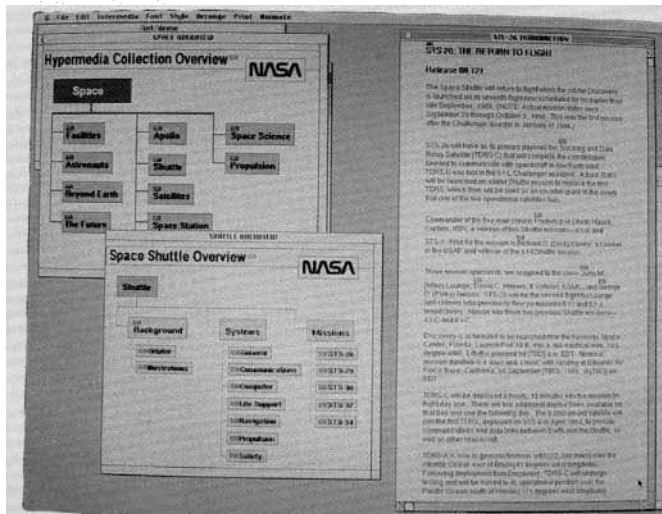
original "hypertext" to emphasize the linking of graphics, animation, sound, and video with textual information. Conklin's survey article [6] describes many of these systems. Some of the more recent systems are also described in the book by Nielsen [19].

The subject of this article is Intermedia, one of these hypermedia systems. The authors were part of a team that developed Intermedia at Brown University's Institute for Research in Information and Scholarship (IRIS) between 1985 and 1990. Intermedia is distinct from many other hypermedia systems in that it is intended to model a multiuser hypermedia framework rather than a single hypermedia program. We did not want to create an isolated hypermedia "island," where linking functionality was limited to connections between homogeneous data managed by a single program [16]. Our intention was to create a model for how hypermedia functionality should be handled at the system level, where linking would be available for all participating applications in much the same way that copying to and pasting from the clipboard facility is supported in the Macintosh and Microsoft Windows environments.

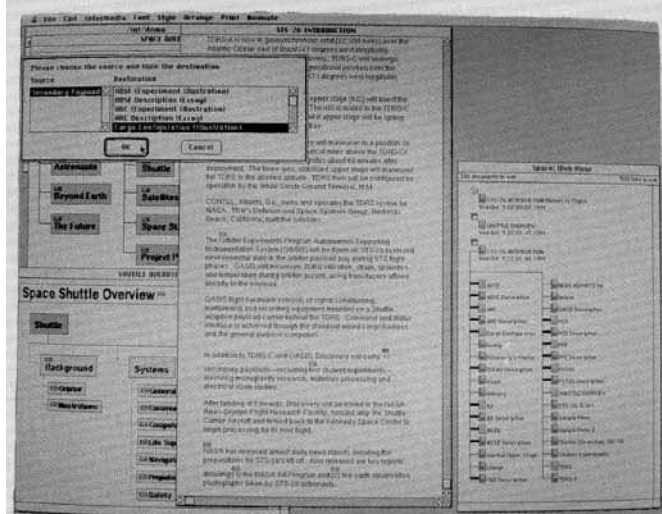
Intermedia presents the user with a graphical file system browser (a functional equivalent of the Macintosh Finder); a set of direct-manipulation editors for text, graphics, timelines, animations, and videodisc data; a browser for link information; a set of linguistic tools; and the ability to create and traverse links between any two selections in any document in the system. As will be explained in greater detail, information about selections (called anchors) and links between these selections is maintained in a database management system (DBMS). This separation of link data and document data is a distinctive feature of the Intermedia design.

The Intermedia user begins by

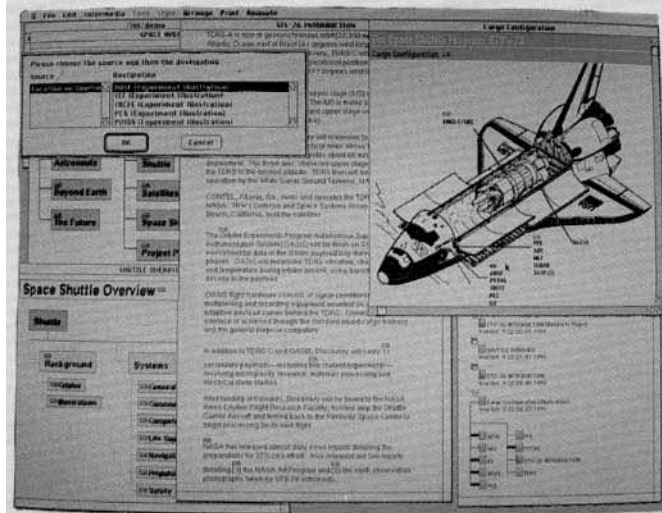
Slide 1



Slide 2

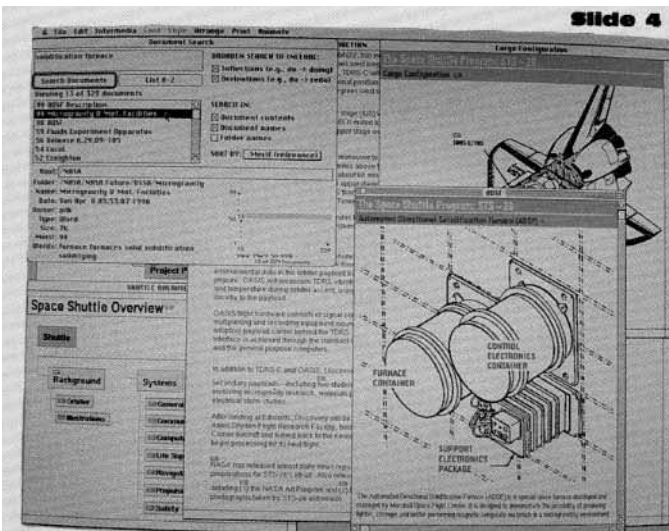


Slide 3



All illustrations used in this article were created by Dynamic Diagrams, Inc.

Slide 4



Slide 1

The user has opened the web for this collection of NASA program documents and followed links from the label about the Space Shuttle in the **SPACE OVERVIEW** diagram in the upper left to **SHUTTLE OVERVIEW**, a diagram describing material on the Shuttle program on the lower left. These overviews are created with the InterDraw graphics editor. From this secondary overview the user has followed a link to **STS-26 INTRODUCTION**, an InterWord text summary of the STS-26 mission. Following the link highlights the destination anchor, in this case the first line of the document.

Slide 2

The user has double-clicked on an anchor marker above the phrase "Discovery will carry 11 secondary payloads" in the text summary. The web view is now visible on the right, presenting a map of the 36 documents linked to **STS-26 INTRODUCTION**. The darkened lines indicate the 16 documents linked to the anchor marker the user has selected. The user chooses the link to follow from a list of document names and anchor explainers.

Slide 3

The user has followed a link to a diagram of the cargo configuration. The map in the web view updates to display the nine documents linked to this diagram. The user chooses the anchor above the names of experiments located in the forward portion of the cargo bay and is given a choice of links to five illustrations.

Slide 4

The user chooses to display an illustration of the Automated Directional Solidification Furnace (ADSF). Seeking to locate more information on "solidification furnace," the user selects the Document Search window and looks for any documents in the NASA collection containing these words. The search locates all documents containing any inflection or derivation of the words, returning a list of over 300 documents sorted with relevance ranks from 1 to 99. By sliding the grey bar in the distribution graph in the lower right of the Document Search window, the user limits the list to the 13 documents with a rank of 50 or higher. The highest-ranking document, **ADSF Description**, is linked to the illustration already displayed and so could have been located by following links. The user selects the second-ranked document from the list, **Microgravity and Mat. Facilities**, and displays information about it including the list of relevant words it contains: *furnace, furnaces, solid, solidification, and solidifying*. The user can open this document by double-clicking on the name in the list.

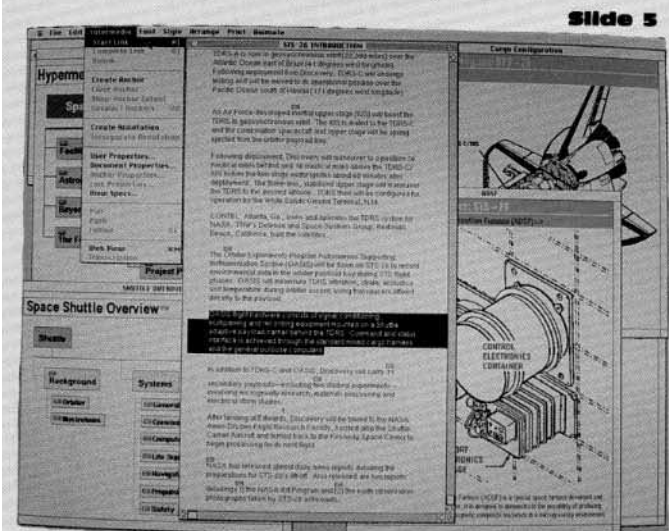
Slide 5

Returning to the **STS-26 INTRODUCTION**, the user selects a paragraph describing the OASIS instrument and creates a new anchor by choosing **Start Link** from the Intermedia menu. This menu manages the anchor and link commands shared by all the Intermedia editors. Information about the new anchor and the pending link is stored by the system in an invisible "linkboard" (analogous to a clipboard) until the user selects a new or existing anchor and completes the link.

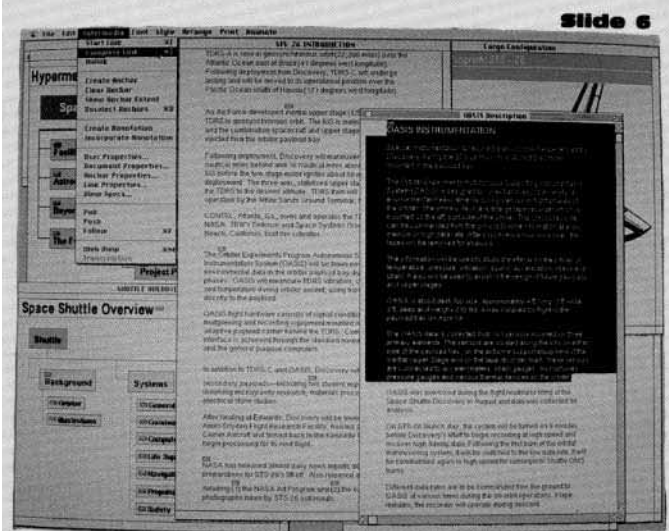
Slide 6

The user completes the link by selecting a new anchor, the title and several paragraphs in **OASIS Description**. The link is forged by selecting **Complete Link** from the Intermedia menu.

Slide 5



Slide 6



A primary goal of Intermedia was to demonstrate that an integrated application environment built using object-oriented programming techniques is the best environment to support hypermedia functionality.

running the Intermedia program from the Unix shell. The shell prompt is replaced by a graphical file system browser that presents documents organized in hierarchical folders. Documents and folders are manipulated using controls similar to those found in the Macintosh user interface. This view of the Intermedia document tree, which resides on a network server, is shared by all workstations on the local-area network. Color slides 1–6 are screen displays showing a user working with an Intermedia collection of nearly 600 documents about various NASA programs.

As Yankelovich et al. have already stated, “Intermedia is both an author’s tool and a reader’s tool. The system, in fact, makes no distinction between types of users, provided they have appropriate access rights to the material they wish to edit, explore, or annotate. Creating new materials and making and following links are all integrated into a single seamless, multi-user environment” [26]. Each Intermedia document opens in its own window and any document can be edited by any user with appropriate privileges. In addition to the usual features offered by direct-manipulation text and graphic editors, Intermedia users can create and follow links between selections in any document presented by the file browser. This link creation and browsing is shared by all parts of the user environment. Again we quote from [26]:

In an effort to fit the link-

making process into a conceptual model already familiar to users, the act of making links between Intermedia documents was modeled as closely as possible on the Smalltalk/Macintosh copy/paste paradigm [10]. If links are to be made frequently, they must be a seamless part of the user interface. In any document, users can specify a selection region and choose the Start Link command from the menu. In any other document, regardless of type, users can define another selection region and choose . . . the Complete Link commands.

To achieve these effects, however, we did not wish to alter an existing operating system or create a new one. We did, in fact, create a hypermedia program running within the Unix operating system to achieve the desired level of integration among different editors. To model how hypermedia would look when integrated at the system level, we created a single application that appears to the user as though it is the entire desktop environment. As we describe in subsection “Integration of Hypermedia into the Desktop,” we believe our strategy can be used as a model for integrating hypermedia into a multiprogram computing environment.

It is not our intention to provide a detailed summary of the features of Intermedia, as this has been done in a number of previous papers. The architecture was first described in [15]. The philosophy of “seamless integration” among application editors found in all ver-

sions of Intermedia and the object-oriented nature of the software development effort are detailed in [26]. Additional features have been described in a number of papers over the past few years. These include features to support temporal data types described in [3, 13, 20]; the integration of morphological services and full-text retrieval into this hypermedia framework described in [7]; extensions to support group annotation of documents described in [4]; and the ability to replicate template structures of linked documents described in [5]. The purpose of this article is to focus on those unique features of the software architecture not previously discussed. We will describe the underlying architecture that made all of Intermedia’s hypermedia functionality possible. The software architecture described in this article is that found in Intermedia 4.0, completed in August 1990.

We have chosen the name *IRIS Hypermedia Services* to describe those parts of Intermedia which make the hypermedia functionality possible. The Intermedia editors, which are responsible for manipulating the content of their own documents, are the surface layer of a complex system. Each of these editors inherits its functionality from the IRIS Hypermedia Services to support hypermedia capabilities.

The overall architecture of the IRIS Hypermedia Services is shown in Figure 1. The Intermedia system appears to the Unix operating system as two processes. The *Intermedia process* is the end-user applica-

In order to accomplish this kind of integration, it is best to establish overall policy goals and then to encapsulate as much of the policy as possible in the object-oriented framework to be used by application developers.

tion. This is built from four distinct layers. Starting from the user's point of view, the first layer consists of the file browser (or Finder), the five editors, each operating on its own document type, and the link browser (or Web View). These each share functionality defined in the second layer consisting of two major building blocks: one for text and one for graphic objects. Above this is the Intermedia Layer, a set of objects for managing hypermedia data that is shared by the building blocks. All these layers, in turn, are made from classes defined by MacApp, an object-oriented application framework developed by Apple Computer.

Intermedia is a hypermedia application written to work on a specific version of Unix (A/UX 1.1) and a specific graphical user interface (Macintosh). The IRIS Hypermedia Services, however, embody those portions of the software that are independent of both operating system and graphical user interface. We feel this portion of the system is a useful model for the way hypermedia services could be migrated into the computing environment and deserves careful consideration by anyone currently designing software with hypermedia functionality.

The IRIS Hypermedia Services is comprised of three parts. The first part consists of the objects defined in the Intermedia Layer and inherited by the end-user applications. The second part is the Link Client, a library that is bound with the Intermedia process. The third

part is the Link Server, a library bound to a DBMS running as a separate process. The Link Client and Link Server communicate via Unix Domain sockets when the two processes are running on the same machine or via Internet sockets when they are running on separate machines.

We will describe the IRIS Hypermedia Services in two stages. The first section of this article, "Features of Hypermedia Policy" describes the hypermedia policy supported throughout Intermedia. Here we will describe the end-user interactions, such as copy and paste of anchors and links, and the application-program responsibilities, such as menu handling, that define what "hypermedia" means in Intermedia.

In the section "The Intermedia Mechanism" the mechanism used to support this hypermedia policy will be discussed. Here we will describe how the "hypermedia" information was managed and integrated with the application data. This will consist of a description of all three parts of the IRIS Hypermedia Services: Intermedia Layer, Link Client, and Link Server.

After the IRIS Hypermedia Services have been described, this article will conclude with a discussion of five of the major issues to be explored in future hypermedia systems.

Features of Hypermedia Policy

A primary goal of Intermedia was to demonstrate that an integrated

application environment built using object-oriented programming techniques is the best environment to support hypermedia functionality. In order to accomplish this kind of integration, it is best to establish overall policy goals and then to encapsulate as much of the policy as possible in the object-oriented framework to be used by application developers. Our intention was to implement all general hypermedia policies at the framework level, leaving only application-specific details unresolved. In this way we assured maximum consistency among Intermedia applications.

Anchor/Link Display, Highlighting and Selection

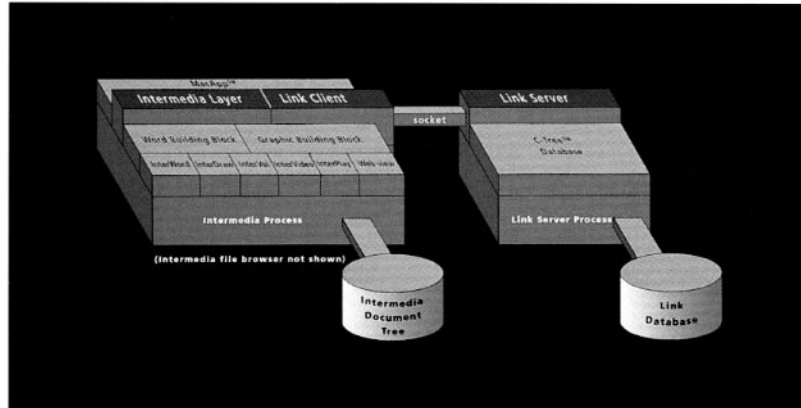
All Intermedia applications must support a persistent selection, called an *anchor*. Each anchor has an *extent*, the application-specific object to which the anchor refers. A *link* in Intermedia is a connection between any two anchors. All applications must support the display of a marker associated with each anchor. This marker has two states: linked and unlinked. A consistent visual appearance of this marker and how it is to be selected by the user is enforced as a matter of general policy in all Intermedia applications.

It is up to the individual applications to determine how to "hook" the anchor to the appropriate document content and how to highlight the extent of the anchor. The selection of an object varies from application to application, depend-

FIGURE 1.

IRIS Hypermedia Services Architecture

This schematic diagram of Intermedia shows the layered architecture of the Intermedia system, with the Intermedia process on the left and the Link Server process on the right. The two processes communicate over a socket connection. The raised and darkened layers identify the portions of the architecture that constitute the IRIS Hypermedia Services.



InterWord Document

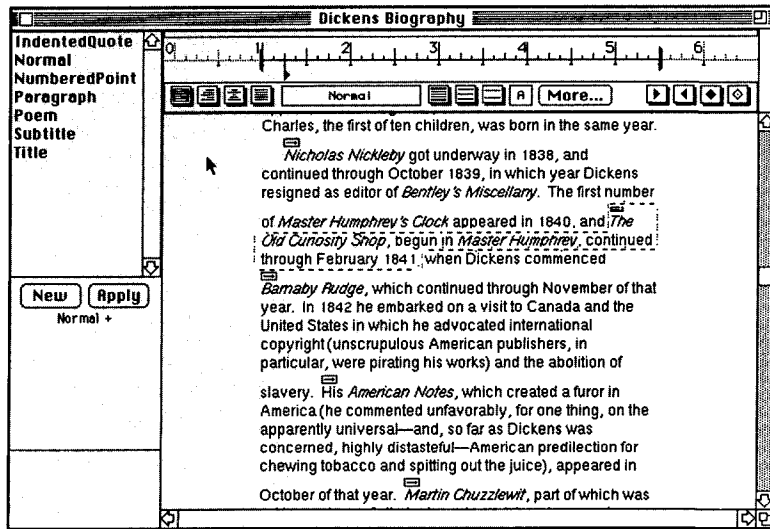


FIGURE 2.

Anchors in Text

Both InterWord (left) and InterMail (right) inherit anchor behavior from the Word Building Block. In these text-based applications, an anchor consists of any contiguous selection of characters. The anchor marker is placed at the upper-left corner of the first character in the anchor and cannot be repositioned. The highlighting of the anchor extent is done with the dotted line of the "marquee."

InterMail Document

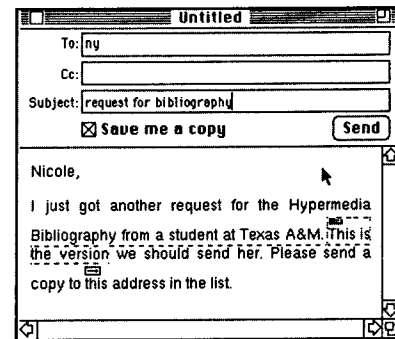
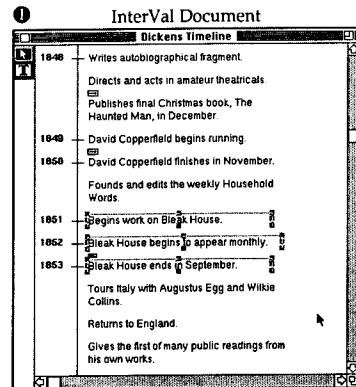


FIGURE 3.

Anchors in Graphics

The remaining four Intermedia applications inherit anchor behavior from the Graphics Building Block. The anchor's extent is indicated by placing grey "handles" around the objects. In the case of the InterVal timeline, any time event or group of events can be an anchor. In InterVideo, any line or group of lines defining a single frame or sequence of frames on a videodisc can be an anchor. Selections do not have to be contiguous, but in both cases the anchor marker is placed at the upper left of the first item selected and cannot be repositioned. In InterVideo the order of selection determines the order in which the video clips are played when following a link to the anchor. Both InterDraw and the InterPlay animation editor allow any graphic object or combination of objects (any object in any frame of the animation in InterPlay) to be an anchor. While at creation the anchor marker appears at the upper left of the first object selected, it can be moved in these two applications in the same way as any other graphic object.



ing on whether the object is text, graphics, or a cell in a table. The relative location of the anchor marker and the anchor extent is also a matter determined by the individual application.

For example, InterWord, the Intermedia text editor, places the anchor marker above the first character in the anchor extent. This marker moves with the anchor as the text is edited, but cannot be repositioned independently of the anchor extent. InterDraw, the graphics editor, and InterPlay, the animation editor, both initially place the marker at the upper-left corner of the first object selected and then allow the user to move the marker to any position in the document. Figures 2 and 3 illustrate how the marker is displayed in the various document types. Figure 2 depicts anchors in text-based applications; Figure 3 illustrates anchors in graphics-based applications.

The relationship between selection of the anchor marker and selection of the anchor extent is a matter of general policy. Regardless of the relative position of the two, selecting the anchor marker constitutes selecting the anchor itself. This means that editing operations (e.g., cut or copy) on a selection which includes the anchor marker affects the anchor and link information. An editing operation on any portion of an anchor's extent not including the marker affects only the content of the anchor.

The user can always clarify what portion of a document is "hooked" to a marker by choosing the marker and displaying its anchor extent.

Data Consistency

It is a matter of general policy to keep document data and anchor/link data consistent at all times. The system must maintain integrity of the links and their respective endpoints in the face of editing operations on document names, locations, and contents. This is necessary to prevent "dangling links," (i.e., link references that point to nonexistent data or empty anchor references that point to data that was not saved).

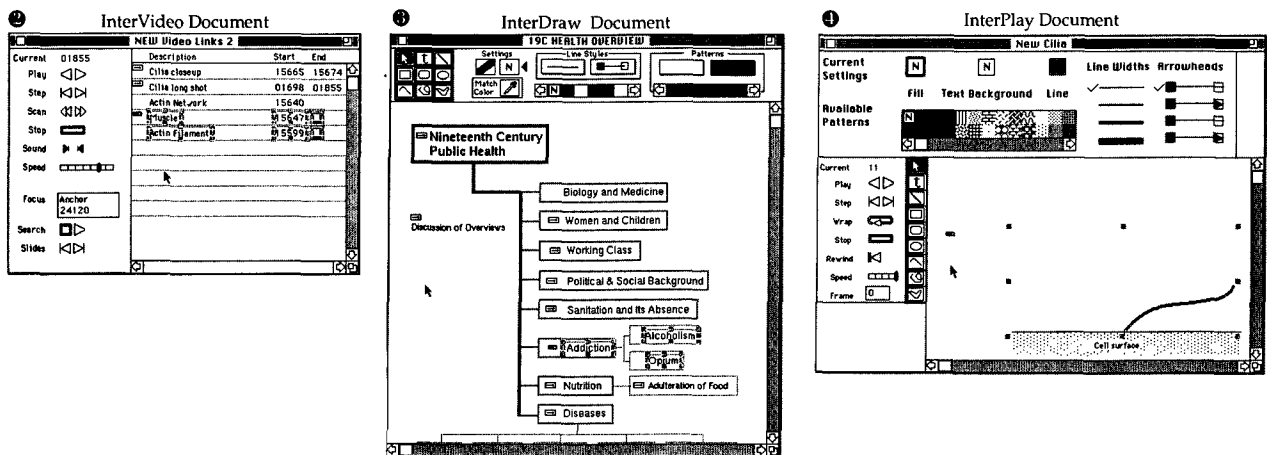
There are basically two mechanisms that have been used to maintain data consistency in hypertext systems. The system can store anchor and link information in the same files as the data, thereby ensuring that editing operations will affect them both. This mechanism is adequate for "read-only" hypertext collections where data consistency is fixed, and has been favored in many commercial systems. However, an editing operation that affects an anchor in one document will not necessarily affect all anchors in other documents linked to that anchor without some further consistency support. Unless some additional system-wide coordination is provided, editing operations as basic as changing a document's name and location will cause "dan-

gling links."

Alternatively, the anchor and link information and the document contents can be stored and managed in a separate but coordinated fashion. This is the mechanism used in Intermedia. We explain the strategy we used in the following two paragraphs, and in greater detail in the subsection "The Intermedia Layer."

Intermedia stores document data in standard Unix files and anchor/link data in a separate database. The Intermedia document tree begins at a "root" on a local or mounted Unix file system such as "/intermedia/documents" and consists of all documents in that file system created by Intermedia editors. The link and anchor information is stored separately in a DBMS. Collections of anchor and link data are partitioned into "webs." From the user's point of view, a web is a specific context in which anchors and links are created and stored. The user opens a web in order to browse and edit a specific collection of links. Each user can open only one web at a time.

Responsibility for coordination of document data with anchor/link data resides in the IRIS Hypermedia Services. Editing operations performed by all applications and the Intermedia file browser pass through this layer of the framework. For example, the deletion of a document from the Intermedia file browser will cause the deletion



of all the anchors and links within and to that document throughout the database. Saving the web, that is saving the changes to the set of anchors and links displayed in the Web View, will cause the application editors to first save editing changes in all documents containing those anchors and links. If the user saves changes to documents but discards changes to the web, anchor and link information that had been cached in memory during the editing session is not written out to the database and remains unchanged.

Selection Handling

It is a matter of policy that all applications manage persistent selections. This must be done by communicating with the IRIS Hypermedia Services, not by saving anchor information in the documents themselves. Each editor must support the ability to edit document data and keep track of the location and current state of the markers and anchor extents. The support for this policy varies from editor to editor.

Menu Handling

All commands used for editing links (create anchor, start link, complete link, etc.), displaying link properties (display anchor properties, display link properties), and browsing links (show anchor extent, follow, push, pull) are consistent across all Intermedia applications. All these commands are managed in the Intermedia Layer rather than by the individual applications and are presented in a pull-down menu shared by all editors.

Editing Anchor/Link Objects

It is a matter of policy that cut/copy/paste operations work in a similar fashion for both document data and anchor/link data. The copying or cutting of the anchor *marker*, not the data in the anchor extent, constitutes the copying or cutting of the anchor and any link data.

Resizing of the anchor extent is

supported in the same way that resizing or extending selections is supported. The undo and redo of editing operations involving anchors and links are also supported in a manner identical with other forms of document data. For example, InterWord, like all Intermedia editors, supports undo and redo of all edit operations in the "live" copy of the document maintained in memory between Save operations. This support is extended to cut/copy/paste of anchor information as well. It is possible to make a selection in one InterWord document that contains text and link markers, and cut that selection, thereby deleting the selected text and unlinking the selected markers. The user can then paste the selection into another document thereby inserting both the text and links. The changes in the Web View map for both documents will be reflected immediately. Undoing the cut operation will affect the "live" copy of the first document while undoing the paste operation will affect the "live" copy of the second document.

Multiuser Access to Documents, Anchors, and Links

Intermedia supports multiple users reading and annotating a single document, and only one user writing to a document at a time. While each user can have only a single web open at a time, any number of users on the same network can simultaneously open the same web and view the same documents, anchors, and links. The first user to edit the document content locks out all other users from editing the content until the document is closed.

By *annotation* we mean the creation or modification of anchors and links as distinct from the creation or modification of document data. Because Intermedia stores anchor and link data in a separate database, we are able to support simultaneous annotation, allowing many users to make links to the same documents at the same time. Intermedia supports separate access rights

for read, write, and annotate privileges on a per-document basis.

This world of multiuser hypermedia running across local-area networks requires a policy establishing when changes to data by one user become available to all users. In Intermedia, all newly created anchors and links are local to the user's workstation until they are saved to disk. Changes made to open documents are not broadcast to other users. Intermedia does not update the view of anchors and links in any open document based on the actions of another user. Each user must close and reopen a document and associated web to see changes made by another user.

Active Anchors

To support editors that manipulate temporal data such as animation and motion video, Intermedia adopted the policy of active anchors [20]. Editors of temporal data support an *action* flag associated with each anchor. The temporal editors must examine the state of the action flag on the anchor when following a link to an animation or video document. Note that active anchors can also be used to execute a query or perform a sequence of actions defined in a script associated with the destination anchor.

If the anchor's action flag is set, a follow into that anchor causes the application-specific action, such as running an animation, to occur. If the action flag is not set, following the link opens the document and highlights the anchor without performing the action. The content of the anchor can be viewed, edited, played, or executed using manual controls.

Transfer of Webs

Sets of linked documents can be transferred from one Intermedia system to another using a canonical form of the link data described in [21]. This transfer functionality supports the selection from the Intermedia file browser of one or more file system folders (representing the document data) containing

one or more web documents (representing the link data).

Though it is not specifically addressed in the Intermedia policy, we recognize that the interchange of hypermedia data between different hypermedia systems is an important issue. One of the authors of this article (Meyrowitz) participated in discussions which led to the Dexter Interchange Format [12], an abstract reference model for hypermedia data. Another author (Riley) participated in the planning process that has led to the proposed HyTime standard (ISO/IEC DIS 10744) [11, 18], a markup language for representing multimedia, hypermedia, and time/space-based documents.

Although Intermedia's transfer policy was originally designed to support the exchange of data between Intermedia users, Killough has shown it to be viable as an interchange format between Intermedia and other hypertext systems as well [14]. This was accomplished by converting the Intermedia interchange data into the Dexter Interchange Format, and from this data generating the format used by the KMS hypertext system [1]. One of the design criteria for HyTime is that it be Dexter-compliant. This suggests it should be possible to convert the Intermedia interchange format into a HyTime format as well.

The Intermedia Mechanism

In Intermedia, the hypermedia policies are encapsulated in a mechanism that has two major components: the *Intermedia Layer* and the *Link Engine*. The Intermedia Layer supports all live data manipulation while the Link Engine supports the storage and retrieval of persistent link data. The storage and retrieval of document data is managed elsewhere in the Intermedia process and is not discussed in this article.

The Intermedia Layer is implemented as a set of classes that inherit from the MacApp application framework as well as newly defined classes that model anchors, links, and webs. The Link Engine is com-

posed of the Link Client, the Link Server, and a DBMS. The Link Client is a class with which the Intermedia Layer communicates. The Link Client in turn communicates with the Link Server. The Link Server is a generalized class communicating data requests to a DBMS.

A schematic diagram of the three parts of the IRIS Hypermedia Services and their relation to the other parts of the Intermedia architecture is shown in Figure 1. This shows how the Intermedia Layer and Link Client are bound in the Intermedia process and the Link Server and DBMS are bound in a separate process.

The Intermedia Layer

The Intermedia Layer consists of extensions to the MacApp framework to support linking capabilities. Included in this layer is a data model class (*IntDoc*) which handles reading anchor/link data associated with a document. A data view class (*IntView*) provides a set of abstract methods for the display of anchor markers and for highlighting anchor extents. Cut, copy, and paste operations are supported by the *IntClipboard* class. Activation of and response to hypermedia operations in the menus (create anchor, start link, complete link, etc.) is supported by methods of the *IntSelection* class.

Within this object-oriented application framework, we created application building blocks that inherit some of the functionality from the Intermedia Layer and override some of its abstract methods. For example, the Graphics Building Block subclasses the *IntView* class to handle graphic object selection and highlighting of anchor extent shown in Figure 4. This, in turn, is used in both the InterDraw and InterPlay editors.

The Intermedia Layer also contains a Web class. When a web is opened, an instance of the class, the Web Object, is created. This object contains a live copy of all newly created and edited anchor and link

data on a per-web basis. This live data is merged by the Intermedia Layer with the current data from the link database. It is here that the policy of data consistency is enforced. The document and link database are synchronized by forcing documents to be saved before the data of the Web Object can be saved. For example, a user has made a link between a selection in an existing document and a selection in a newly created document. To save this link, the user saves the Web View, which first saves the documents that have been edited before saving the anchor and link information in the web. This ensures that all anchor and link information in the link database is synchronized with the current saved state of each document.

Consistency is also maintained in operations that affect anchor and link information in unopened webs. For example, when a document is deleted, all links containing anchors in that document are removed from the database. The delete operation will affect links in any web in the database. In addition, a user can edit a portion of a document that is all or part of an anchor extent in an unopened web. The effect this has on the anchor extent is resolved when the edited document and the web containing that anchor are opened.

Intermedia provides an orientation feature called the Web View which contains a list of the documents viewed by the user and a map of documents linked to the currently active window. The Web View rationale and functionality is described in [23]. The link map in the Web View relies on the Web Object for its information.

All Intermedia editors read in anchor information from the link database and correlate this to the application data in each document at the time the document is opened. Subsequent changes in the Web Object are displayed in the live copy of the document and Web View on each user's workstation, but are not propagated to the live copy of the

same documents elsewhere on the network.

The Link Engine

The Web Object of the Intermedia Layer communicates the anchor,

link, and document transactions to the Link Engine and reads that information from the Link Engine via

the methods of the Link Client class. These methods, which constitute Intermedia's linking protocol,

FIGURE 4.

Application Framework Architecture

The class hierarchy of Intermedia's application framework is illustrated in part. The View method in the MacApp layer is subclassed by *IntView* in the Intermedia layer where several abstract methods to manage the display of anchors and markers are added. These methods are then defined in the *gView* and *tView* classes of the Graphics and Word Building Blocks. In addition to the subclasses from MacApp, the Intermedia layer adds new classes for Anchor, Link, and Web, shown on the right.

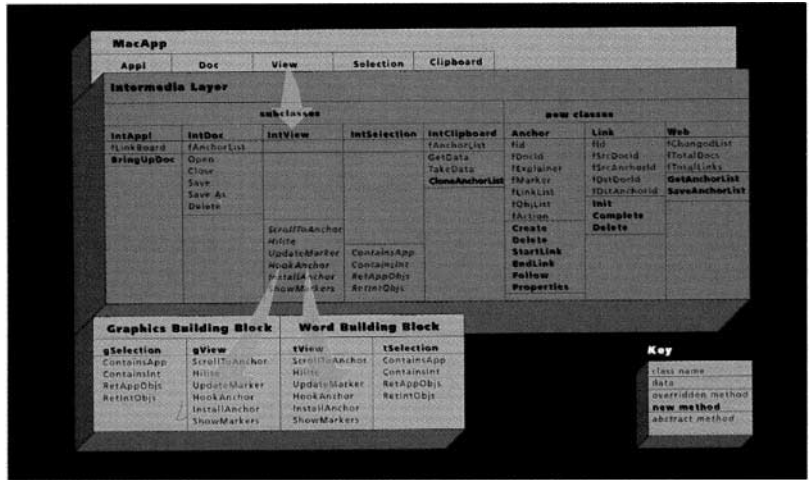


FIGURE 5.

Client/Server/DBMS Flow of Control

The *GetAnchorList* method in the Web class of the Intermedia Layer (see Figure 4) requests a list of all anchors in a document. This request is passed to the *AnchorGetFirst* method in the Link Client ①. This passes the request over the socket connection to the abstract method *AnchorGetList* in the Link Server ② which is defined in Link Ctree method *AnchorGetList* ③. This method translates the request into a series of low-level DBMS operations (REDVREC), requesting all anchors with a specific document ID, returning the anchors and repeating until done ④. This list of all anchors is cached in the Link Server ⑤ and then returned to the *AnchorGetNext* method in the Link Client ⑥ one at a time across the socket.

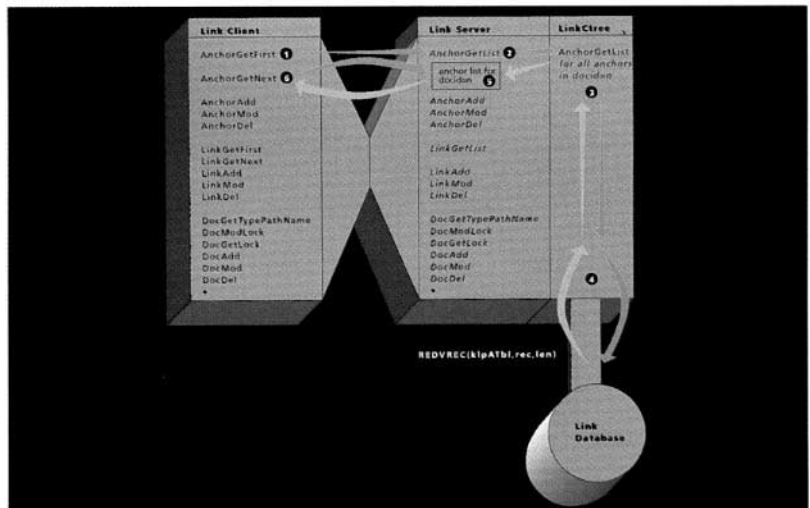
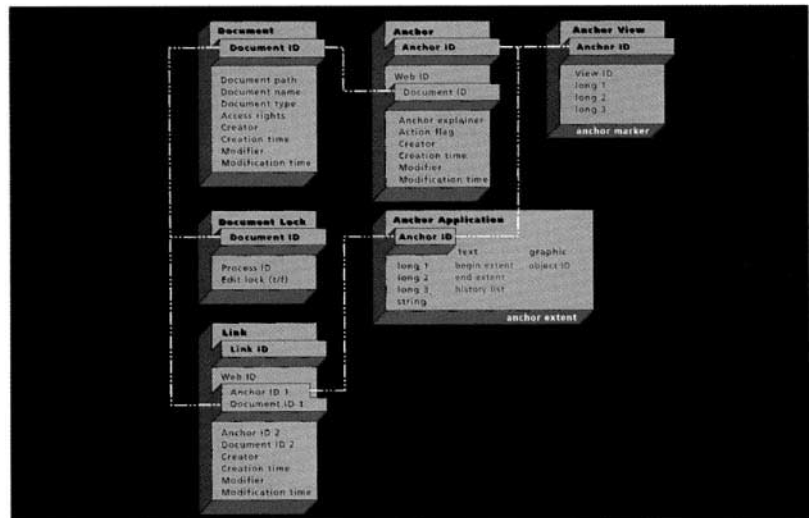


FIGURE 6.

Link Engine Data Model

This shows the six major relations in the Link Engine data model. The primary key for each relation is raised to illustrate how the data is connected. Note that the Document ID joins the Document relation to the Document Lock, Link, and Anchor relations. The Anchor ID joins the Anchor, Anchor View, Anchor Application, and Link relations. The Link is defined as a field of the Anchor and Link relations. The Web is defined as a field of the Anchor and Link relations.



support establishment of connections over sockets, opening and closing of databases, and a set of operations for manipulating document, link, and anchor data.

An example of how this data is communicated will illustrate the way different parts of the architecture share in the hypermedia mechanism. When the Intermedia process is initialized, the Link Client establishes a connection to the Link Server and opens the link database. When the Intermedia user opens a web document, the Web Object in the Intermedia Layer requests all the corresponding web information from the Link Engine. The Web Object has a method called *GetAnchorList* (shown in the Web class in Figure 4) which returns a list of all anchors and links for a document each time a document is opened. When the user opens a document, this request is communicated to the Link Client, which passes the request to the Link Server over the socket, using a predefined set of byte codes. The Link Server accepts these byte-coded messages and then performs a series of operations on the database to retrieve all anchor and link information for that document by making calls to a DBMS, in this case the commercially available *C-Tree* [9]. The information is then returned to the Intermedia Layer along the same communication path. This is shown in Figure 5, which follows the *AnchorGetFirst* method from the Link Client, to the Link Server, to the DBMS, that performs the low-level operation to read the anchor data for the specified document ID from the database. The resultant list of all anchors is cached by the Link Server. This list is then accessed one anchor at a time by the Link Client through the *AnchorGetNext* method. The Intermedia Layer then passes the application-specific anchor information to the document editor by invoking abstract methods that the document editor has overridden. These methods then display a marker for each anchor and hook

the anchor to the appropriate data in the document.

The Link Engine can be described in terms of a data model and the operations that can be performed on that data model.

Data Model

The persistent link information is organized in five major relations: document, anchor, anchor view, anchor application, and link. A schematic diagram of these relations is found in Figure 6.

Document Relations. The *document* relation matches a unique document identifier with the document name, Unix path and the basic properties such as creator, creation time, modifier, and modification time. The primary index for this relation is the unique document identifier to support efficient retrieval and reliable operation independent of changes in document name and location.

A secondary relation is the *document lock* relation which maintains information on which documents are locked for editing. The unique document identifier for each locked document is related to the process identifier of the editing process. This information is used during release of document locks.

Anchor Relations. The *anchor* relation matches a unique anchor identifier with a document identifier and basic properties of the anchor including creator, creation time, modifier, modification time, and explainer. The primary index for this relation is the unique anchor identifier.

Additional anchor information is stored in two other relations.

The *anchor view* relation is used to store the location of the anchor markers in graphics documents where the marker is positioned independently from the anchor extent. This relation, which contains an anchor identifier, a view identifier, and three long integers, was designed to support applications that present more than one view of

the document data. An example of this was InterSpect, a viewer for three-dimensional wire-frame models that was part of an early version of Intermedia. InterSpect supported both a two-dimensional and a three-dimensional view of the same data.

The third relation with anchor information is the *anchor application* relation. Intermedia applications store various amounts of information to define an anchor extent in this relation, which contains an anchor identifier, three long integers, and a string. The string, which was included in the design to support extensibility, is unused. In the InterDraw application, for example, an anchor can consist of more than one graphic object. The identifier for each graphic object in an anchor extent is stored in one of these relations. Managing information about changing anchor locations in text streams presents a particular challenge. The InterWord application stores the information it needs to resolve the location of each anchor—consisting of beginning offset, ending offset, and an index into a document's history list—in a single instance of the *anchor application* relation. The history list, a sequence of numbers appended to the end of the document, tracks additions and deletions to the document. By indexing to the appropriate point in the history list, InterWord is able to resolve the appropriate location of each anchor.

Link Relation. The *link* relation matches a unique link identifier with basic properties of the link including creator, creation time, modifier, and modification time. Each link relation consists of a pair of unique anchor identifiers and unique document identifiers for each side of the link. The primary index for this relation is the unique link identifier.

Other Relations. The database also contains an *object identifier* relation which provides the next available

unique identifier for the major object types. The numbers generated by this relation are used to identify all new document, anchor, and link objects.

Finally, the database contains database-computed queries stored in separate relations that contain the scope information displayed by the Web View. These relations contain total numbers of anchors, links, and documents for each unique web identifier.

Link Engine Operations

The initial operation of the Link Engine is the opening and closing of a socket connection between the Link Client and the Link Server. The locations of the Intermedia document hierarchy, the Link Server, and the link database are passed as environment variables at run time. Once the connection is established, the Link Engine can open and close the database, using the DBMS bound with the Link Server. Another basic operation returns the next available unique identifier assigned to documents, links, and anchors.

The Link Engine supports similar sets of operations on the data associated with documents, anchors, and links. These including adding, modifying, and deleting entire objects or data elements associated with each object. Move and delete operations are supported for folders (i.e., Unix directories) as well. Separate methods for manipulating access rights, path name, and edit lock are supported for documents.

As discussed previously in the subsection "Transfer of Webs," a set of operations for exporting and importing document, anchor, and link information is also provided. Exporting one or more folders creates a copy of the document hierarchy in those folders and a set of ASCII files in a canonical form containing the link data for webs in those folders. The importing of document and link data in this format is supported by a complementary set of operations.

Features for the Next Generation

In designing and implementing Intermedia, we have identified a number of features that are critical for making hypermedia a useful part of the computing environment. The development of the Intermedia system ceased before we could implement solutions for these issues. Some of these features, such as wide-area hypermedia support, require a more extensive networking architecture than we have employed. Others, such as the need for filtering tools, are features that were in the original specification for Intermedia (see [25] for an early discussion of filtering), but were never implemented.

The following discussion represents the results of our efforts to define future hypermedia system capabilities that took place at IRIS over the past several years. The next generation of hypermedia systems should explore solutions to these issues.

Integration of Hypermedia into the Desktop

If hypermedia functionality is isolated in specific hypermedia programs, the usefulness of that functionality will be very limited. Hypermedia functionality must be a feature of the entire computing environment, fully reflected in whatever graphical user interface integrates information on the user's screen. Whether that interface is built around a desktop, notebook, or room metaphor, hypermedia linking must be available in all documents, in all applications.

This means the Link Engine, an equivalent of the Link Client/Link Server/DBMS layer in Intermedia, should be as integral to the computing environment as the file system is today.

The functionality found in the Intermedia Layer, however, should be separated into two parts: high-level integration support for applications and an intermediary process that handles link-related requests.

Integration support can be provided through an application program interface (API) to a high-level toolkit, requiring the application developer to make calls to the API at the appropriate times. The toolkit should contain the embodiment of a system's hypermedia policies. In addition, a predefined set of callback routines will have to be implemented by each application. An alternate method of providing this integration support is an application framework such as Apple's MacApp.

Between the application features found in this API and the database features found in the Link Engine, a general desktop hypermedia system will require an intermediary process we call a Link Hub. This Link Hub should serve four primary functions:

- keeping the list of pending links (i.e., links not yet committed to the database);
- managing the creation of links;
- knowing how to follow a link (i.e., knowing how to locate documents and which application to invoke in order to open them);
- providing link information to a link browser equivalent to the Intermedia Web View.

By employing an API, a Link Hub, and a Link Engine, all applications should be able to incorporate linking, allowing hypermedia to be as common an integrating feature as cut, copy and paste is today.

Multiple Webs

The limitation of having only one web open at a time has proven too restrictive. We created the concept of a web in Intermedia to provide a specific context in which to collect all related anchors and links. This context helped to separate sets of anchors and links overlaid on the same documents for different purposes. Restricting the user to one active web simplified the user interface.

Developers should keep in mind that providing *no* context, that is displaying all links at all times, may seem adequate when hypertext

linking is limited within an isolated application or even a specific collection of documents managed by a single application. However, this lack of context will prove inadequate when hypertext linking is spread across all applications. We urge designers of future hypermedia systems to face the challenge of managing multiple contexts.

It is clear from our experience with Intermedia that being able to view more than one set of links at a time is quite useful. For example, hypermedia functionality is commonly used today to support on-line documentation and "help" systems. Users would obviously like to be able to browse links in such a help system while browsing the links in other webs of information. It would also be useful to have a private web open for containing personal links and annotations in a set of documents while browsing through a public web associated with the same documents.

There are additional complexities in the application's user interface and in the Link Engine that arise in this case. When allowing multiple webs to be open, it becomes more complicated to indicate where links should be stored when they are created. Likewise, requesting from the Link Engine all link information for a specific document or the equivalent of a web view will require more complex database operations.

Once multiple active webs are supported, it is also reasonable to expect users to need links that explicitly connect not only the documents but, in effect, the webs themselves. In the same way that hypermedia links locate and open documents, future systems should support links that locate and open entire contexts of links. An example might be a link that not only opens a document, but also adds another collection of links to the user's view, effectively opening another web.

Filtering Tools

In an environment with many users

and large numbers of links, it is necessary to provide tools to locate the most useful information in a timely fashion. This should be done by providing tools to limit the kinds of anchors and links displayed in a document, which we call *exposure filtering*, and to identify the anchors and links that meet a specific criteria, which we call *collection filtering*. Both of these features were part of the original design of Intermedia, but neither was fully implemented.

In a sense, the Intermedia web is a simple exposure filter. The suggestions for implementing multiple webs contained in the preceding subsection, "Multiple Webs," would be one step in a strategy for adding filtering to a hypermedia system. However, filtering by web is too *coarse* to be effective. For example, a web may be created by the collaboration of 10 authors, but a reader may only want to see the links made by two of them. The filtering must be finer than entire webs. To filter out links that are not pertinent, the user should be able to formulate a simple query based on the properties of the anchors and links. Application of this query should expose only those anchors and links which match the query. Browsing through documents, the user should only be shown anchors and links that passed through the filter.

Using the same query mechanism, the user should also be able to collect a list of anchors and links that match a query. From this list the user should be able to go immediately to the relevant information or further refine the query.

Additional functionality would be gained by supporting user-defined attributes attached to anchors, links, and documents. Filters could be applied to match combinations of these attributes. These filters could be stored and shared among users.

Any of these features would require additional database management support. The Link Engine would have to support queries consisting of arbitrary combinations of attributes associated with docu-

ments, anchors, and links.

Wide-Area Hypermedia

The multiuser hypermedia described previously in the subsection "Multiuser Access to Documents, Anchors, and Links" is based on supporting links in documents accessible across a local-area network (LAN) model using Sun's Network File System (NFS), a networking scheme available on most Unix workstations. As similar file system support becomes available for a wide-area network (WAN), it is imperative to also add support to hypermedia functionality. To provide a WAN hypermedia system, there are several requirements:

- documents must be accessible across the WAN;
- documents must be uniquely identified across the WAN in a more manageable way than pathnames/filenames; and
- link data must be represented in a more distributed fashion.

To maintain link information across a WAN, the notion of a document identifier must be extended beyond the Intermedia model. Intermedia document identifiers are unique on a per-volume (file system) basis. For WAN operation, document identifiers must be unique on a network-wide basis. Additionally, information about the physical location of the document would have to be stored in a more general fashion. We recommend a link database per volume, with anchors identified at the volume/document level. Essentially, the only identifier specification that would need to change from the Intermedia database schema is the document identifier. A service on each machine could find a particular document by querying the machine at the WAN destination and asking for the document at volume identifier x with document identifier y . Each file system could have a service that mapped volume identifier/document identifier pairs to local file system handles.

Once a scheme for storing

unique document identifiers is established, the most difficult question that remains is how to distribute the link information. The Intermedia architecture assumed the anchor information was kept in a per-volume, multiuser database and the link relation existed in the same database and connected two items on the same volume. A WAN extension would want to keep all *anchor* information in a per-volume, multiuser database while distributing the link information. Anchor information only concerns persistent selections within a document, and so this information should be managed with the volume on which the document is stored.

Link information in a WAN, multivolume world may connect anchors in documents that exist on two different volumes, on two different machines. This calls for a modification in how and where the link information is kept. We suggest keeping the *link* information in a per-volume database, but *replicating* it, so that the link information is stored in the databases for the volumes in which the anchors/documents at either side of the link reside.

Transparent Object Storage

Since Intermedia is based on an object-oriented framework, the most logical DBMS to use for storing the anchor and link information would be an object-oriented database (OODB) as was suggested in [22]. This was not done due to the lack of a viable commercial OODB systems at the time. However, we feel this option should be seriously considered as OODB technology continues to evolve.

Combining an OODB with a transparent object-oriented interprocess communications mechanism would allow the Link Engine we have described to be considerably simplified. Much of the work done by the Link Engine involves flattening the data objects managed by the Intermedia Layer into a form that can be transferred across the network and stored in a con-

ventional database. Whether that was a full relational database management system such as *INGRES* or a B-tree database system such as *C-Tree*, the anchor and link information had to be converted into a series of database management operations and associated arguments.

In recent months we have worked with Sun Microsystem's Remote Procedure Call (RPC) functions and developed an object-oriented interprocess communications mechanism we call RPC++. This mechanism supports sending messages to remote objects as well as passing objects across the network. Combining a facility such as RPC++ with an OODB would reduce the load on the Link Engine and provide a hypermedia system with transparent object storage.

Conclusion

Our vision of hypermedia is based on a belief that the computing environment should be able to reflect the ever-changing connections inherent in all information. Bush [2] envisioned how computing machines could be used to support intellectual work and contain the complex interlinkage of accumulated knowledge found in the human record. The "docuverse" envisioned by Nelson [17] would contain all of the world's literature, accessible and interlinked from any point on the global network. The cooperative work environments created by Engelbart [8] were shared electronic spaces where groups of workers could manage and exchange concepts and ideas in the form of linked structures. We feel Intermedia has been a stage in the development of a computing environment better able to support intellectual work.

We cannot allow the current market, with its millions of stand-alone personal computers, to transform this global hypermedia vision into a set of isolated applications. For hypermedia to work in today's multimedia platforms in the large, the function of making and travers-

ing links must find its way out of the individual application and into the shared space of the desktop environment where we do our daily work. Attention must be focused on how to make hypermedia available across all applications, as well as across WANs.

We believe that the IRIS Hypermedia Services represents a useful design for those researchers and developers who are concerned with hypermedia in the large. The development of Intermedia has demonstrated that hypermedia functionality can be implemented in an object-oriented application framework. We believe our architecture has demonstrated the value of separating the anchor and link data from the document data and using a database management system for storing and retrieving the link information. We believe a similar design in which the hypermedia functions are well integrated into the computing environment will provide the basis for the next generation of desktop computing. **□**

References

1. Akscyn, R.M., McCracken, D.L., and Yoder, E.A. KMS: A distributed hypermedia system for managing knowledge in organizations. *Commun. ACM* 31, 7 (July 1988), 820-835.
2. Bush, V. As we may think. *Atlantic Monthly* 176, 1, 101-108.
3. Catlin, T.J.O. and Smith, K.E. Anchors for shifting tides: Designing a 'seaworthy' Hypermedia system. In *Proceedings of the 12th International Online Information Meeting* (London, England, Dec. 6-8, 1988) Oxford and New Jersey: Learned Information, 1988, pp. 15-25.
4. Catlin, T., Bush, P.E., and Yankelovich, N. InterNote: Extending a Hypermedia framework to support annotative collaboration. In *Hypertext '89 Proceedings* (Pittsburgh, Pa. Nov. 5-7, 1989) ACM, N.Y., 1989, pp. 365-378.
5. Catlin, K.S., Garrett, L.N., and Launhardt, J.A. Hypermedia templates: An author's tool. In *Hypertext '91 Papers* (San Antonio, Tex. Dec. 15-18, 1991).
6. Conklin, J. Hypertext: An introduction and survey. *IEEE Comput.*

- 20, 9 (Sept., 1987), 17–41.
7. Coombs, J.H. Hypertext, full text, and automatic linking. *International Conference on Research and Development in Information Retrieval (SIGIR '90)*. (Brussels, Sept. 5–7, 1990).
 8. Engelbart, D.C. and English, W.K. A research center for augmenting human intellect. In *AFIPS Conference Proceedings, 1968 Fall Joint Computer Conference*. (San Francisco, Calif., Dec. 9–11, 1968), AFIPS Press, Montvale, N.J., pp. 395–410.
 9. FairCom. *C-tree File Handler Programmer's Reference Guide*. Columbia Mo, 1988.
 10. Goldberg, A. *Smalltalk-80: The Interactive Programming Environment*. Addison Wesley, Reading, Mass., 1984.
 11. Goldfarb, C.F., and Newcomb, S.R. ANSI Project, X3.749-D, 'Hypermedia/Time-based Document Structuring Language (HyTime)'. X3V1.8M/SD-7.
 12. Halasz, F., and Schwartz, M. The Dexter Hypertext Reference Model. In *Proceedings of the Hypertext Standardization Workshop*. J. Moline, D. Benigni, and J. Baronas, Eds., NIST Special Publication 500-178, Gaithersburg: National Institute of Standards and Technology, Jan. 1990, pp. 95–134.
 13. Kahn, P., and Haan, B.J. Video in hypermedia: The design of Inter-Video. *Visual Resource*, VII (1991), 353–360.
 14. Killough, R.L. Hypertext interchange with the Dexter Model: Intermedia to KMS. Texas A&M University, Dept. of Computer Science, Aug. 1990.
 15. Meyrowitz, N. Intermedia: The architecture and construction of an object-oriented Hypertext/Hypermedia system and applications framework. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '86)* (Portland, Ore. Sept. 29–Oct. 2, 1986).
 16. Meyrowitz, N. The missing link: Why we're all doing Hypertext wrong. In *The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information*, E. Barrett, Ed. MIT Press, Cambridge, Mass. 1989, 107–114.
 17. Nelson, T.H. *Literary Machines*. Swarthmore, Pa., T.H. Nelson, 1981.
 18. Newcomb, S.R., Kipp, N.A., and Newcomb, V.T. The 'HyTime'

- Hypermedia/Time-based Document Structure Language. *Commun. ACM*, 34, 11 (Nov. 1991), 67–83.
19. Nielsen, J. *Hypertext and Hypermedia*. Academic Press, San Diego, Calif., 1990.
 20. Palaniappan, M., Yankelovich, N., and Sawtelle, M. Linking active anchors: A stage in the evolution of Hypermedia. *Hypermedia* 2, 1 (1990), 47–66.
 21. Riley, V. An interchange format for Hypertext systems: the Intermedia model. In *Proceedings of the Hypertext Standardization Workshop*, J. Moline, D. Benigni, and J. Baronas, Eds., NIST Special Publication 500-178, Gaithersburg: National Institute of Standards and Technology, Jan. 1990, pp. 213–222.
 22. Smith, K.E., and Zdonik, S.B. Intermedia: A case study of the differences between relational and object-oriented database systems. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '87)*. (Orlando, Fla., Oct. 4–8, 1987).
 23. Utting, K., and Yankelovich, N. Context and orientation in Hypermedia networks. *ACM Trans. Inf. Syst.* 7, 1 (Jan. 1989), 58–84.
 24. van Dam, A. Hypertext '87 keynote address. *Commun. ACM*, 31, 7 (July 1988), 887–895.
 25. Yankelovich, N., Meyrowitz, N., and van Dam, A. Reading and writing the electronic book. *IEEE Comput.* 18, 10 (Oct. 1985), 16–30.
 26. Yankelovich, N., Haan, B.J., Meyrowitz, N., and Drucker, S.M. Intermedia: The concept and the construction of a seamless information environment. *IEEE Comput.* 21, 1 (Jan. 1988), 81–96.

CR Categories and Subject Descriptors: C.3 [Computer Systems Organization]: Special-purpose and Application-based Systems; H.3 [Information Systems]: Information Storage and Retrieval; H.4.3. [Information Systems]: Information Systems Applications—*Communications Applications*

General Terms: Design

Additional Key Words and Phrases: Hypermedia, hypertext, Intermedia, IRIS Hypermedia Services

About the Authors:

BERNARD J. HAAN is currently project manager of the Multimedia Group at Siemens-Nixdorf Information Systems in Cambridge, Mass. His current research interests include desktop inte-

gration of hypermedia and multimedia. **Author's Present Address:** Siemens-Nixdorf Information Systems, 4 Cambridge Center, Cambridge, MA 02142; email: bhaan@sni-usa.com

PAUL KAHN is research scientist and project manager at Brown University's Institute for Research and Scholarship. His current research interests include the design and presentation of digital information. **Author's Present Address:** Institute for Research and Scholarship, Brown University, Box 1946, Providence, RI 02912; email: pdk@iris.brown.edu

VICTOR A. RILEY is a technical staff member in the System Software Technology Center of the Advanced Systems Division at Silicon Graphics, Inc. His current research interests include hypertext and hypermedia document interchange, and the standardization involved with interchange, as well as object-oriented programming and databases, and window systems. **Author's Present Address:** Silicon Graphics, Inc., 2011 North Shoreline Boulevard, P.O. Box 7311, Mountain View, CA 94039-7311; email: var@sgi.com

JAMES H. COOMBS is a research scientist at Brown University's Institute for Research in Information and Scholarship. His current research interests include the integration of advanced searching techniques into the daily working environment. **Author's Present Address:** Institute for Research and Scholarship, Brown University, Box 1946, Providence, RI 02912; email: jhc@iris.brown.edu

NORMAN K. MEYROWITZ is director of Advanced Software Technology at GO Corporation. His current research interests include object-oriented building blocks, next-generation user environments, hypermedia, compound documents, text processing, user-interface design, and object-oriented programming. **Author's Present Address:** GO Corporation, 950 Tower Lane, 14th Floor, Foster City, CA 94404; email: nkm@go.com

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© ACM 0002-0782/92/0100-036 \$1.50