Workspaces: The HyperDisco Approach to Internet Distribution

Uffe Kock Wiil

Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg Øst, Denmark
Email: kock@cs.auc.dk

John J. Leggett

Center for the Study of Digital Libraries
Department of Computer Science
Texas A&M University
College Station, Texas 77843-3112, USA
Email: leggett@cs.tamu.edu

ABSTRACT

Hypermedia concepts are currently being deployed in a variety of information systems such as the World Wide Web, software development environments, large engineering enterprises, collaborative authoring systems, and digital library systems. The complex requirements of these application areas have resulted in extensive research into hypermedia infrastructures.

The HyperDisco project is about design, development, deployment and assessment of hypermedia infrastructures. Previous HyperDisco experiments have dealt with integration of a small set of tools supporting authoring and extension of the integrated tools to support multiple collaborating users and multiple versions of shared files. These experiments were conducted on a local area network using a single centralized workspace. The latest version of HyperDisco supports collaboration and versioning over multiple workspaces distributed across the Internet. This paper gives a brief overview of HyperDisco, describes the workspace concept and reports on the latest experiments: (1) an experiment that allows the use of multiple workspaces on a local area network, (2) an experiment that allows workspaces to be distributed across the Internet, and (3) an experiment focusing on hypermedia modeling and presentation issues of distributed workspaces.

KEYWORDS: Distributed workspaces, hypermedia infrastructure, Internet distribution, open hypermedia system, link replication, name service, hypermedia presentation

1 INTRODUCTION

Hypermedia concepts are currently being deployed in a variety of information systems such as the World Wide Web (WWW) [3], software development environments [1, 42], large engineering enterprises [11, 22], collaborative authoring systems [32] and digital library systems [6, 29]. The complex requirements of these application areas have resulted in extensive research into hypermedia infrastructures.

As indicated in Figure 1, the current research on open hypermedia systems (OHS) has its roots in two fairly independent threads of research: link server systems (LSS) and hyperbase management systems (HBMS). Research on HBMS began more than a decade ago as reported in the "Hypertext Abstract Machine" paper at Hypertext '87 [5]. By 1990 several other research groups began to publish HBMS research results. Five research groups have been active in the HBMS field for the past 5-10 years: GMD-IPSI [2], Aarhus University [11], University of North Carolina [30], Texas A&M University [20], and Aalborg University [38]. Research on LSS was first discussed at Hypertext '87 when Meyrowitz handed out his much quoted paper: "The missing link: Why we are all doing hypertext wrong" (later published in [24]). The Hypertext '89 paper on "Sun's Link Service" [26] was quickly followed by papers on PROXHY [18], Microcosm [13] and Multicard [28]. More recently systems like Chimera [1], Hyper-G [23] and KHS [27] have emerged.

The focus of LSS approaches is to develop middleware [4] components that can provide hypermedia linking functionality to tools orthogonal to their storage and display functionality. Using the services of a LSS, existing tools in the computing environment can be "hypermedia enabled" – thus providing hypermedia linking to and from information managed by the tools without altering the information itself. Driven by the idea of a unified hypermedia environment, HBMS research has been concentrating on providing support at a more basic level. In addition to hypermedia linking functionality, HBMS also provide support for storage. HBMS have been used to develop middleware components that allow existing tools to use both linking and storage functionality (if desirable). HOSS [25] is going a step further in its attempt to add hypermedia functionality to the default operating environment by developing a set of hypermedia operating system services.

Hypermedia versioning research started mostly as an integrated part of HBMS research. The first hypermedia versioning papers were published at ECHT '92 [12, 41]. By 1994 the growing subgroup of hypermedia versioning researchers started to arrange their own line of events. At the same time people from the HBMS and LSS areas got together and formed the OHS community. Not surprisingly, OHS can be identified as originating from either the LSS or the HBMS thread of research. Mutual influences of the LSS and HBMS

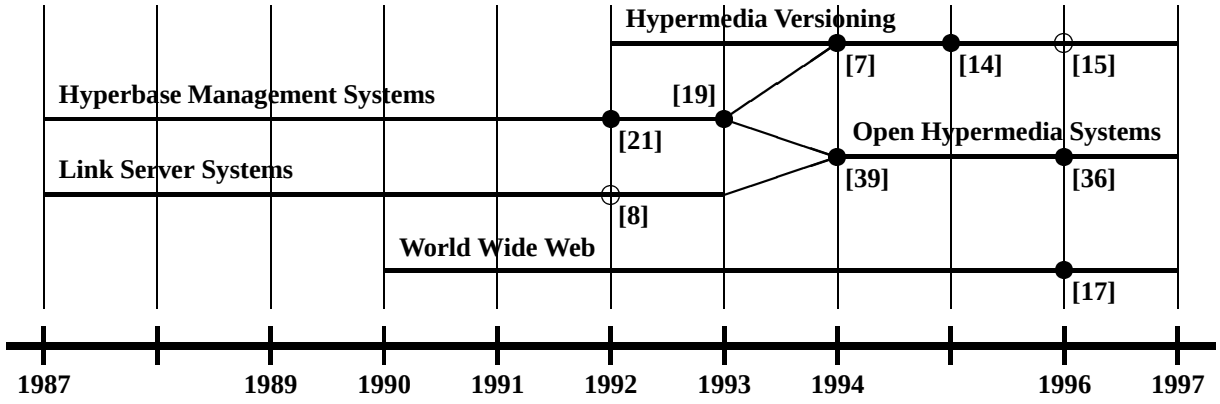

Figure 1: A historical view of hypermedia infrastructure research. Hypertext conference years are marked in the time line at the bottom. Important subgroup events (workshops and panels) at hypertext and closely related conferences are marked as bullets (workshops) and circles (panels).

approaches are closing the gap between these two approaches to OHS. HBMS are now being deployed as part of open, extensible and distributed hypermedia system architectures and the HBMS themselves typically provide open, extensible data models that allow existing tools to be integrated. LSS are currently providing support for storage, collaboration and versioning. The enormous success of the WWW has also had its impact on hypermedia infrastructure research, with the latest trend addressing scalability and distribution issues (number of hyperbases, amount of storage, number of concurrent users, etc.) [13, 20].

1.1 HyperDisco Overview

HyperDisco [37] is an example of an OHS that is based on a HBMS. HyperDisco is the latest research prototype in a long history of hypermedia infrastructure research at Aalborg University that started in 1988. Previous research prototypes include the HyperBase [34] / EHTS [35] system and the Hyperform hypermedia system development environment [38].

The HyperDisco project is about design, development, deployment and assessment of hypermedia infrastructures. The work merges previous experiences and results from existing lines of hypermedia infrastructure research (Figure 1). The overall goals are: (1) to work towards innovative new hypermedia infrastructures for integration and extension of tools and (2) to try to influence the development of future generations of hypermedia infrastructures for the Internet. HyperDisco provides a flexible infrastructure for integration and extension of (third-party) tools. The specific objectives in the design and development of the HyperDisco prototype were to place the characteristics of the prototype on the high end of the following seven dimensions: scalability, openness, distribution, heterogeneity, interoperability, extensibility and computation [37].

To meet these ambitious objectives, a layered, extensible and object-oriented OHS infrastructure was developed as the core of the prototype [37]. This infrastructure allows tools to be treated differently in the integration process: (1) tools can be integrated at different tool-dependent levels and make use of as many (or as few) of the provided hypermedia services as desirable and (2) there is no single model of integration to which all tools must adhere. Each tool (or tool group) can have its own specialized model of integration and its own specialized protocol for accessing hypermedia services.

A number of experiments have been performed with HyperDisco to assess and test the limits of the infrastructure. Previous use and experimentation can be grouped into four major experiments each exploring and validating an important feature of HyperDisco [37].

Simple integration models. The purpose of the first experiment was to test the integration features available in HyperDisco and to experiment with different simple (anchor-link) integration models. Emacs [31] was integrated at various levels using different integration models.

Integration and extension models. The second experiment focused on more advanced (data-wrapper node) integration models. Again, Emacs was integrated at various levels using different integration models. Based on the results of the first two experiments, a particular data-wrapper node integration model was selected for the two tool extension experiments described below.

CSCW extension. In this experiment a small selection of tools supporting authoring was integrated: Epoch [9], LaTeX (a document processing system), PageView (a postscript viewer), Xdvi (a dvi viewer), dvips (a conversion script) and a printer script. With Epoch as the central tool, the integrated authoring environment was extended to support multiple users collaborating on a shared set of files.

Versioning extension. This experiment added versioning capabilities to the collaborative authoring environment. Epoch allows users to maintain and retrieve various versions of the contents of the shared set of files.

These experiments were all conducted on a local area network (LAN) using a single centralized HBMS (called *workspace* in HyperDisco). The HyperDisco infrastructure was running

with multiple tools and tool integrators. Although several workspaces existed, participating tools were only capable of using a single workspace.

This paper reports on the latest HyperDisco experiments addressing the issues of supporting multiple workspaces distributed across the Internet.[1] Section 2 explains the workspace concept and compares it to related work. Sections 3 through 5 each focus on a particular experiment that deals with important issues relating to distributed workspaces. Section 3 describes the effort that went into extending HyperDisco from a single workspace to multiple workspaces on a LAN; Section 4 deals with the extension of HyperDisco to allow workspaces to be distributed across the Internet; and Section 5 presents an experiment focusing on hypermedia modeling and presentation issues of distributed workspaces. Section 6 concludes the paper and outlines future work.

2 THE HYPERDISCO WORKSPACE CONCEPT

The HyperDisco infrastructure provides two distinct layers of functionality. The *integration model* provides hypermedia linking services including a flexible linking protocol. The *data model* provides basic hypermedia storage services. The HyperDisco infrastructure is composed of distributed workspaces, tool integrators and participating hypermedia and third-party tools as depicted in Figure 2.

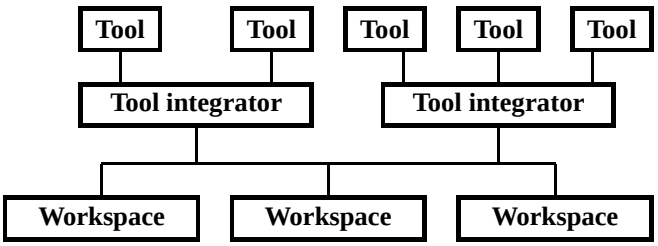

Figure 2: The HyperDisco infrastructure (from [37]). Tool integrators provide the integration model and workspaces provide the data model.

A workspace is an autonomous HBMS serving as a gateway to a set of multimedia files residing either in the underlying file system or in the HBMS itself. In addition to storage and retrieval of file contents and file handles, workspaces offer a wide range of hypermedia services to participating tools ranging from simple services such as anchoring and linking of files to more advanced services such as access control on files and links, support for collaborative authoring of files and support for multiple versions of files.

Each workspace maintains a list of legal users that are allowed to access the hypermedia services. Legal users do not need to have an account on the workspace host machine, but must simply be listed as legal users of the workspace. Workspaces can be private, public or belong to a group of users. A hypermedia researcher could for example setup a number of private workspaces for personal research and teaching material, a number of group workspaces (e.g., one for each collaborative project), and one or more public workspaces to publish research material on the Internet. A public workspace grants access to any user.

Tool integrators allow distributed heterogeneous tools to be integrated and control access and operation of multiple workspaces. A typical single user session with one tool, one tool integrator and multiple workspaces will include the following simplified steps:

1. The user starts a tool integrator on his local machine.

2. The user starts a HyperDisco enabled tool which connects to the tool integrator using the HyperDisco protocol.[2]

3. Based on the particular requests made by the tool, the tool integrator opens connections to one or more workspaces to provide the necessary services. Workspace connections are kept open until the end of the session.[3]

4. The session is terminated when the tool disconnects and closes its connection to the tool integrator. This causes the tool integrator to close its connections to workspaces used in the session.

5. The user may decide to terminate the tool integrator.

Many of the important issues and mechanisms involved in this kind of session spanning multiple distributed workspaces will be addressed in Sections 3 through 5.

A growing number of hypermedia infrastructure projects are addressing the many difficult issues involved in Internet distribution. These different approaches can be divided into two categories, those that build on concepts and mechanisms of the WWW and those that develop Internet distribution infrastructures that run independent of the WWW. Both types of approaches are important and have their justification. While it is important to see how far the concepts and mechanisms of the current dominating hypermedia Internet infrastructure (the WWW) can be taken, it is equally important to see how far we can get if we go beyond the current limitations of the WWW and create new solutions.

The first category is by far the largest and includes approaches like Hyper-G [23], Microcosm's Distributed Link Service [13], KHS [27], the Distributed Graph Storage (DGS) system [6] and Chimera [33]. The latter category includes the Texas A&M approaches (HB3 [20] and HBproc in HOSS [25]), Microcosm The Next Generation (TNG) [13] and HyperDisco.

We will compare HyperDisco to two of these approaches (one from each category) throughout the paper: (1) the WWW [3] since this approach is the common denominator in most of the Internet distribution projects and at the same time by

[1] HyperDisco experiments typically add to the overall functionality of the infrastructure. Thus, the versioning and collaboration facilities introduced in earlier experiments are carried forth into the experiments on distributed workspaces.

[2] The HyperDisco protocol is a session-oriented network service based on tcp/ip (sockets). The socket connections are kept open for the duration of a session. The same network protocol is used both between tools and tool integrators and between tool integrators and workspaces.

[3] The tool integrator will have at most one connection to each workspace, since workspace connections are shared among connected tools.

Name	Location	Host name	Port	Process ID
"default"	"/home/kock/Hyperform/bin/default/"	"dauphine.ics.uci.edu"	5000	13524
"hyperdisco"	"/home/kock/Hyperform/bin/hyperdisco/"	"opera.ics.uci.edu"	5000	3022
"designspace"	"/home/kock/Hyperform/bin/designspace/"	""	0	0
"test"	"/home/kock/Hyperform/bin/testing/"	""	0	0

Table 1: HyperDisco name service information. In this example, the "default" workspace is running on "dauphine," the "hyperdisco" workspace is running on "opera" and the "designspace" and "test" workspaces are currently not provided. The "Location" is used to specify where essential workspace information is stored. The "Process ID" is used by **stop-workspace** to terminate services.

far the most used hypermedia Internet infrastructure.[4] (2) HBproc since this approach is to our knowledge the only other approach that provides a distributed hypermedia workspace concept (independent of the WWW) similar to the one implemented in HyperDisco with support for versioning, collaboration, etc. We will summarize our findings and round off the related work discussion in the conclusion.

3 MULTIPLE WORKSPACES ON A LOCAL AREA NETWORK

The support for multiple workspaces was completed early in the HyperDisco project (1994), but has never been documented or used (except for various test settings). The tool integrator was selected to be the controlling instance and was extended with a Workspace class. The workspace (HBMS) only required minor updates, since distribution control is almost completely encapsulated in the tool integrator. Workspaces register with a name service during initialization and unregister from the name service when terminating services (see Table 1). Based on the name service information, the Workspace class implements the following operations: **change-workspace**, **start-workspace**, **stop-workspace**, **create-new-workspace**, **move-workspace**, **rename-workspace** and **delete-workspace**. When starting tool integrator and workspace instances, it is possible to specify which name service to use (a default name service is used if nothing is specified). By supporting multiple name services, several HyperDisco instances can be running concurrently and in isolation from each other.

Each tool integrator has a default workspace that stores its integration model classes. A particular workspace can be specified as the default in the command line at invocation if desired. The tool integrator will always be connected to its default workspace and will monitor changes to its integration model classes via events. If an integration model class is updated (from some other tool integrator) the tool integrator will load and install the changes. Each participating tool has a current workspace to which the tool integrator sends requests from the tool. A tool can use **change-workspace** to change its current workspace; this may trigger a **start-workspace** call to provide the new workspace. Thus, the tool integrator might have connections to several workspaces at one time. The remaining workspace operations are provided for the systems administrator to maintain the operation of the system at runtime.

With the infrastructure in place for multiple workspaces, the main task in this experiment was to extend the data model and integration model classes to allow links to cross workspaces. Since HyperDisco links are n-ary and first class objects, this immediately raises a number of issues that must be addressed. Where should across workspace links be stored? In one of the workspaces (say the current workspace of the user creating the link)? In a dedicated link workspace that manages all across workspace links? In each of the workspaces that participate in the link?

The WWW stores links together with the source file, making it difficult to support *back* links. Supporting back links in the WWW would require a query involving all servers in the world to determine if any of these have links that point to a particular destination file and the query would have to be repeated whenever a new file is loaded. On a global scale this solution is impossible due to the large number of servers. The solution could probably work on a LAN with a small number of workspaces, but the queries will be relatively slow and might result in an incomplete list of links due to inaccessibility of workspaces.[5]

[4] Since the WWW is in a tremendous flux, it is necessary to elaborate on the particular configuration of the WWW that is used. HyperDisco is compared to the following commonly used WWW configuration: Apache WWW server, HTTP protocol, HTML format and Netscape Navigator 3.0. CGI scripting is a powerful way to extend the standard configuration on the server side of the WWW. HyperDisco provides a powerful scripting mechanism as well and can also be extended in numerous ways in both the tool integrator and in the workspace [37]. We will only compare the standard features of the WWW and HyperDisco and consider the scripting mechanisms to be outside the scope of the comparisons.

[5] One reviewer noted that the WWW already has a mechanism in place for supporting back links without querying all servers in the world. Index servers like AltaVista use Web crawlers to gather information about file contents and create indices that are available on the Internet. Back links could be supported by making AltaVista queries using a URL (uniform resource locator) as the search criteria. This solution has two problems that minimizes its applicability even for the WWW. Firstly, the search results can in the best case only be approximate, and secondly, the solution does not work in real time, since the browsing tools will depend on accessibility and response time of index servers like AltaVista every time a new file is loaded. Since the suggested solution assumes that links are binary and has no notion of anchors, link types and contexts, it is no good for systems like HyperDisco and HBproc.

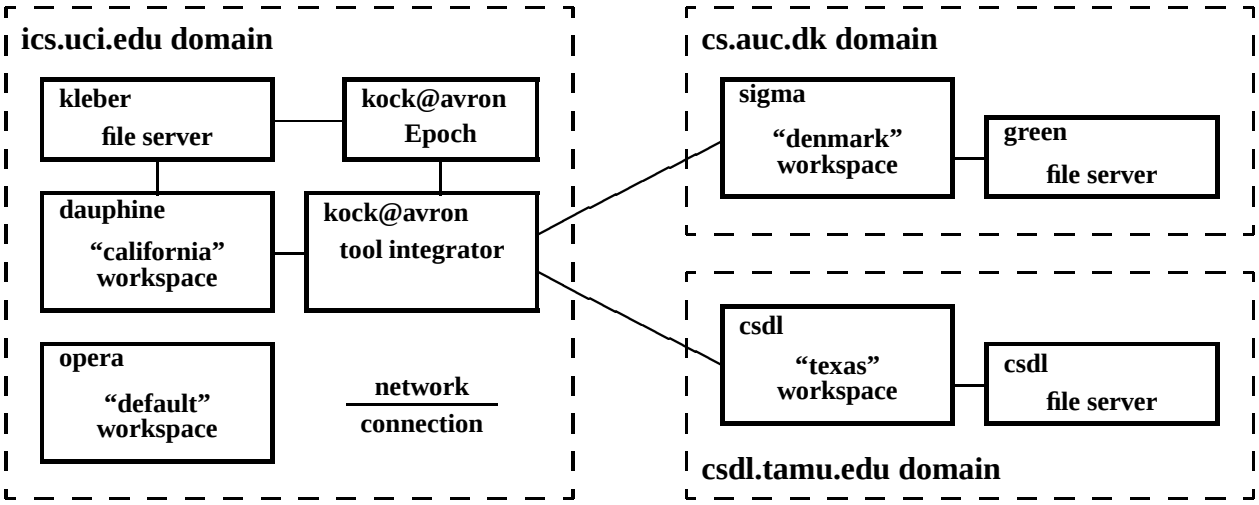

Figure 3: An example scenario with four workspaces ("default", "california", "denmark" and "texas") running at different Internet domains.

Replication of links across participating workspaces raises the issue of maintaining link integrity in operations that create, update and delete links. HyperDisco implements this solution. A link in HyperDisco contains a number of endpoints, each consisting of a triple *(workspace name, node identifier, anchor identifier)*. The tool integrator Link class maintains integrity in link operations that involve multiple workspaces. Before performing link operations such as create and delete, the Link class ensures that all workspaces connected by the link are reachable, and based on the outcome, either continues or cancels the operation. The **follow-link** operation is treated differently, since inaccessibility of workspaces does not violate integrity. The Link class queries the current workspace for link endpoints and opens as many of these endpoints as possible. This solution is less vulnerable to workspace failures and promotes autonomy of the individual workspaces, since all the information necessary to follow a link is stored in a link instance in each workspace connected by the link. Autonomy is less important when running on a LAN, since it is possible for a single person to control all the workspaces. Preservation of autonomy is *crucial* when workspaces are running at different Internet domains with no single person in control (see Section 4). The solution introduces additional overhead in link operations that span multiple workspaces (except for **follow-link** operations). Thus, the price for n-ary links (which is beneficial in the browsing process) is paid during the authoring process – this actually makes a lot of sense when looking at the WWW where "surfers" greatly outnumber authors.

In contrast to HyperDisco's link replication solution, n-ary links in HBproc (called associations) are stored in the current context selected by the user. Contexts can be private, public or shared and can be stored in any hyperbase. Associations can point to components in different hyperbases and these hyperbases can be running anywhere on the Internet. HBproc provides a general solution to the multiple hyperbase problem introducing two levels of link clustering, contexts (data modeled by composites) and hyperbases. Like HBproc, HyperDisco supports multiple contexts (composites) in each workspace, but none of the integrated tools have used this feature. From the tools' point of view each workspace provides a single context. Future experiments will address this issue and allow tools to make use of multiple contexts in each workspace (see Section 6).

4 INTERNET DISTRIBUTION OF WORKSPACES

The move from LAN distribution to Internet distribution can best be characterized as a move from a controllable local setting to a global setting based on autonomous domains. Figure 3 depicts a single-user setting with multiple workspaces distributed across the Internet. User "kock" has started an Epoch editing session and is running a tool integrator on his private workstation "avron" with "california" as the default workspace. Within that session "kock" is able to operate on files locally and at remote domains. Inspired by the WWW, HyperDisco was extended with **get-file** and **put-file** operations to introduce file location transparency. Before a file is loaded, HyperDisco determines the location of the file. If the file is stored locally, Epoch will use the normal load and write functions, otherwise it will use **get-file** and **put-file** to transport files from and to the remote domain. In this setting, "kock" can create and follow links between files, load files, and update files located at different domains.

Other users need only to start a tool integrator and participating tools to join the scenario. For example, user "leggett" (located in the csdl.tamu.edu domain) can join "kock" in a synchronous collaborative authoring session by starting a tool integrator on his private workstation with "texas" as the default workspace and his favorite integrated editing tool. During the session "kock" and "leggett" can use the collaborative authoring capabilities of HyperDisco (introduced in earlier experiments [37]) to edit files located in different workspaces running at different Internet domains.

This experiment revealed two important issues: (1) the need

Name	Location	Host name	Port	Process ID
"default"	""	"opera.ics.uci.edu"	5000	0
"california"	"/home/kock/Hyperform/bin/california/"	"dauphine.ics.uci.edu"	5000	5241
"texas"	""	"csdl.tamu.edu"	5000	0
"denmark"	""	"sigma.cs.auc.dk"	5000	0

Table 2: "kock"'s name service in the ics.uci.edu domain used in the scenario depicted in Figure 3. "default" is the default workspace for the entire ics.uci.edu domain. "default" is a local, public workspace that runs on "opera" and registers with the domain name service. "california," "texas" and "denmark" are group workspaces owned by "kock" – they register with "kock"'s name service in their local domain (e.g, "california" registers with "kock"'s name service in the ics.uci.edu domain and "texas" registers with "kock"'s name service in the csdl.tamu.edu domain).

for a name service that supports workspace location transparency and (2) the need for globally unique file names. The URL (uniform resource locator) concept introduced by the WWW is a general solution to the problems of file and workspace location and naming on a global scale, but does not handle the transparency issue [16]. URNs (Uniform Resource Name)s have been proposed for this purpose and are currently under study [16].

HyperDisco uses a two-level mapping scheme inspired by the URL concept. At the system level, workspace names are mapped to Internet hosts and ports using the name service. At the user level, files are uniquely identified by a file name (which is stored in the *file* attribute of the node) and a workspace name. In this experiment, the name service was updated to emphasize autonomy of both users and workspaces, to distinguish between local and remote workspaces, and to distinguish between private, group and public workspaces. The name service is no longer centralized. Each user has their own private name service. Local workspaces automatically maintain their Host name, Port and Process ID in the owner's name service. Users are responsible for maintaining Host name and Port information for all public and remote workspaces that they wish to access (see Table 2). Name and Location information can still be maintained by **create-new-workspace**, **move-workspace**, **rename-workspace** and **delete-workspace**. **start-workspace** and **stop-workspace** only work on local workspaces that are owned by the user.

The HyperDisco solution is different from the WWW solution in at least three ways. Firstly, users can assign their own names to existing workspaces. If desirable, workspace names can be identical to the server part of a URL by using the Host name and Port (e.g., "opera.ics.uci.edu" or "opera.ics.uci.edu:5000"). Secondly, users only have to supply workspace information (Name, Host name and Port) once in a name service. In the WWW the entire URL must be specified every time an outgoing link is created. Thirdly, users can maintain and use different name services for different purposes: one for browsing the workspaces on the Internet, one for personal work, and one for each collaborative project. This allows users to cluster existing workspaces into logical subdomains to create a better view of the potentially large number of workspaces.

Except for one major difference the HyperDisco solution is similar to the HBproc solution. HBproc uses a distributed name service (called the Server Information Manager or SIM). There is one SIM per machine. Each SIM has information about some arbitrary set of hyperbases. Hyperbases register with the SIM on the machine on which they are running. SIMs may be arbitrarily networked together and propagate hyperbase information amongst themselves. For name services, tools contact the SIM on the machine on which they are running. In contrast to HBproc, HyperDisco puts a burden on the user to maintain part of the name service information. HBproc avoids this by propagating information between name services that are networked together. This issue is quite complex since it has two counteracting factors, scale and transparency. The WWW solution is highly scalable, but does not address the transparency issue well. The HBproc solution provides complete transparency but requires a solution with the complexity of DNS (Unix Domain Name Service). The solution in HyperDisco is an attempt to take the best of both approaches by providing a high degree of transparency and placing minimal requirements on the users.

5 PRESENTATION OF DISTRIBUTED WORKSPACES

The previous six experiments have mostly focused on bringing the distributed workspace functionality into the integrated tools and have to a large extent ignored issues of presenting the functionality to the users. This experiment was completely devoted to enhancing the workspace concept and the way the concept is presented to users through the user interfaces of participating tools.

Figure 4 shows a screen dump of XEmacs [40] when running as an integrated HyperDisco tool. XEmacs is partially integrated, which (in HyperDisco terms) means that XEmacs stores its files in the file system and is capable of managing anchors and links both to and from file contents. XEmacs can also respond to HyperDisco events. The HyperDisco extension is written entirely in Emacs lisp code that is loaded at invocation time using a command line option: *xemacs -l hyperdisco.el <file name>*. The extension attempts to provide additional functionality that is similar to existing functionality in XEmacs in its look and feel.

XEmacs supports n-ary links, allows multiple links from each anchor and maintains positional anchors in the text. Posi-

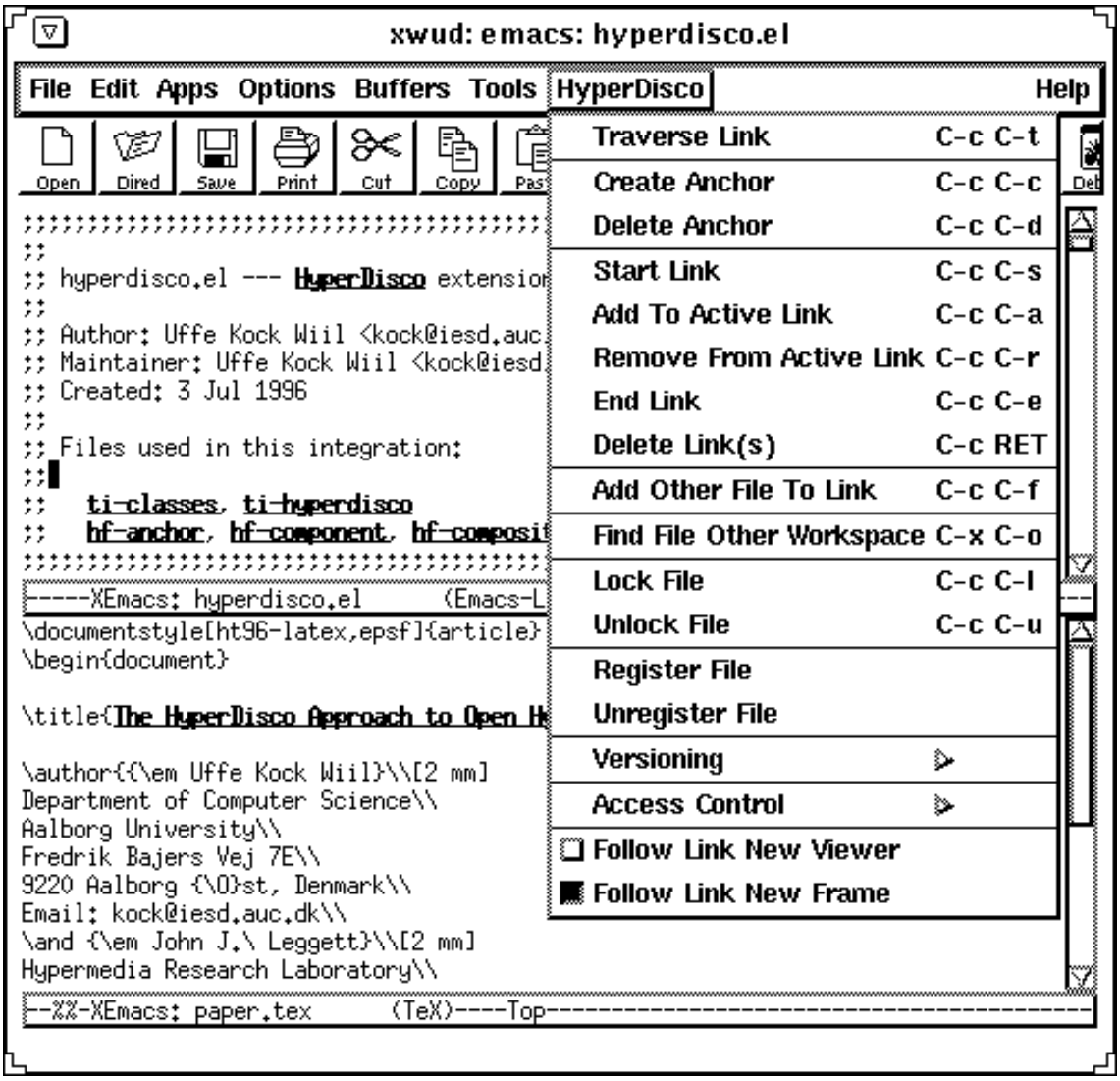

Figure 4: XEmacs as a participating HyperDisco tool. Anchors are bold-faced and underlined. HyperDisco operations are available from the keyboard and through the "HyperDisco" menu. In addition, links can be followed by double-clicking on the anchors.

tional anchors are based on the "extent" and "face" capabilities in XEmacs. Multiple files from different workspaces can be displayed at the same time. For example, the two files displayed in Figure 4 are located in different workspaces, respectively the "california" workspace (hyperdisco.el) and the "denmark" workspace (paper.tex). Each file has its own buffer that maintains a number of "buffer-local" variables such as workspace name, file name, node identifier and anchor list. Files can be registered with HyperDisco explicitly by invoking a command from the menu (see Figure 4). Files will automatically be registered when anchors are created in the file.

Links are created in the following manner. The user creates an anchor in a file (by selecting a piece of text and invoking the "Create Anchor" function). The user then selects the "Start Link" function. If the cursor is positioned over an anchor, it will automatically be added to the new link. Otherwise the user can add the anchor by positioning the cursor over the anchor and invoking the "Add To Active Link" function. The user can now create other anchors and add more endpoints to the active link. XEmacs uses the buffer-local variables to compose the endpoint data (workspace name, node identifier, anchor identifier), which relieves the user of remembering file and workspace location. The tool integrator stores the endpoint data until the "End Link" function is invoked. "End Link" creates a link instance in each workspace connected by the link as mentioned earlier.

This experiment attempts to provide a flexible and tailorable interface to the distributed workspace functionality as illustrated by the following three examples. In the first example, the menu allows the user to tailor the behavior of XEmacs when following a link. Two options can be set: "Follow

Link New Viewer" and "Follow Link New Frame." The first option specifies whether the tool integrator will start a new instance of the tool or send follow link events to an existing instance when opening XEmacs endpoints. If the first option is not set, the second option specifies whether a follow link event results in opening up a new frame displaying the endpoint or reusing the current frame by displaying two buffer windows in the frame (as in Figure 4). No matter which options are set, XEmacs displays (scrolls to) and highlights destination anchors. In the second example, the menu allows the user to add endpoints in files managed by non-integrated tools. The "Add Other File To Link" function queries the user for a workspace name, a file name and a tool name.[6] This information is used to register the file (if necessary) and add a whole-component endpoint (void anchor identifier) to the active link. When following the link, the tool integrator will start the tool and the tool will display the file. No hypermedia capabilities are available in the tool. In the third example, the menu allows files to be loaded from any accessible workspace. The XEmacs "Find File" function works for files located in XEmacs' local domain, while the "Find File Other Workspace" function works for files located in workspaces running at other Internet domains. "Find File Other Workspace" works just like "Find File" except that, in addition to querying the user for a file name, it also queries for a workspace name.

6 CONCLUSION

This paper has addressed a number of issues in the development of hypermedia infrastructure that supports content versioning and collaborative authoring of files located in autonomous workspaces distributed across the Internet. HyperDisco provides: (1) unique file names across workspaces, (2) link replication to support n-ary links across workspaces, and (3) location transparency of both files and workspaces through a two-level name service.

We conclude by summarizing the comparisons of HyperDisco to the WWW and to HBproc made throughout the paper.

HyperDisco and the WWW. The WWW provides a hypermedia interface for using services and browsing information on the Internet. The most important contribution of the WWW is the emphasis on simplicity, extensibility and autonomy which has resulted in a highly scalable and massively distributed system. The emphasis in HyperDisco has been to develop a hypermedia infrastructure for global information management systems. HyperDisco has many features not found in the WWW: (1) third-party tools can be integrated to browse and author information, (2) some integrated tools can be extended to support versioning of file contents and collaborative authoring of file contents and hypermedia structure (anchors and links), (3) links are first class objects (separated from the linked sources), making authoring of hypermedia structure a matter of "pointing and clicking" in the user interfaces of integrated tools (as illustrated with XEmacs), (4) links are n-ary, and (5) access control is provided both at the workspace level and at the file/link level. The workspace concept is believed to be highly scalable with respect to users, workspaces and storage, but HyperDisco has not been tested in large-scale settings that resemble the WWW for obvious reasons.

HyperDisco and HBproc. HyperDisco and HBproc both provide a concept of workspaces (hyperbases) with support for Internet distribution, versioning, collaboration, access control and across workspace links. The fact that HyperDisco and HBproc have many features in common comes as no surprise, since these two projects have been running concurrently for several years. The mutual inspiration of the projects have resulted in many common project goals and in two infrastructures that are addressing the same key issues. It should however be noted that the projects represent independent threads of research that naturally have achieved their comparable features in different ways using different techniques. The comparisons made throughout the paper identify a number of differences in the way these two infrastructures implement the underlying mechanisms of the distributed workspace concept such as name service and across workspace links. While a complete comparison is beyond the scope of this paper, a closer examination of HBproc and HyperDisco reveals that the protocols, data models, system architectures and integration mechanisms underlying these two infrastructures are quite different.

The latest experiments in the HyperDisco project provide evidence that the global hypermedia infrastructure of the future can provide many additional features not presently found in the WWW. Combining the features and underlying technologies of the WWW, HBproc and HyperDisco would result in a global information management system that supports browsing and authoring of information by individuals as well as groups. It will be a major challenge to develop such a system. The HyperDisco project has approached this challenge by developing a hypermedia infrastructure that runs in parallel and in isolation from the WWW. Other approaches focus on: (1) interoperability between existing hypermedia systems and the WWW (e.g., [2], [13] and [27]), (2) extending the WWW with state-of-the-art hypermedia features (e.g., [6] and [33]), and (3) developing reference models for future hypermedia infrastructures (e.g., [10] and [43]). Hopefully, all these efforts in combination will result in major improvements in future global hypermedia infrastructures.

6.1 Future Work

We will continue to improve and experiment with the HyperDisco infrastructure. Our current plan includes the following topics.

[6] Inspired by Chimera [1], HyperDisco implements a process invoker table storing information on how tools must be initiated. The table uses the application and file attributes of the nodes. Integrated tools such as XEmacs store both the tool and file name when a file is registered in HyperDisco. When registering files maintained by non-integrated tools we could avoid prompting the user for the tool name by maintaining a mapping between file types and tools. For flexibility reasons, we leave it up to the user to decide which tool should display a particular file instead of always using the same tool for all files of a certain type (e.g., vi to display text files and pageview to display postscript files). By recognizing a file as a pair (tool, file), a file can be registered multiple times by different tools, thus allowing for multiple views on files (like Chimera). From a user's perspective there is no difference between the solutions in Chimera and HyperDisco. From a modeling perspective there is a noticeable difference, since Chimera provides views as first-class objects and HyperDisco does not.

Multiple Views. We plan to further enhance support for multiple views on accessible information. This will be achieved by adding access control capabilities to anchor objects. When loading a file in this setting, users can only access anchors that they have created and anchors on which they have read permissions. This can be combined with a filtering mechanism that allows users to specify different subsets of the accessible anchors to be displayed (e.g., all private anchors, all anchors that belong to a certain group, etc.).

Composites. HyperDisco supports composites, but none of the participating tools have used this feature. Currently, tools use workspaces to cluster information. We will experiment with different ways to use composites to provide clustering of information within workspaces. Clustering is yet another way to provide multiple views.

Multimedia. So far HyperDisco has focused on tools that support text. In the future we wish to integrate and extend tools that support other types of media.

Interoperability With the WWW. True interoperability requires equal participation of both systems, which means that both systems have to be updated. Updating HyperDisco is not a problem, but we currently do not find it feasible to spend time updating the WWW. When the write capability (the "put" function) becomes a natural part of the WWW, we will look closely at different ways to provide interoperability between HyperDisco and the WWW.

Scalability and Heterogeneity. These problems have actually been addressed quite well by the WWW, which runs on multiple hardware platforms and operating systems allowing millions of users to concurrently access enormous amounts of storage on millions of servers. The ongoing activity of using HyperDisco as a basis for digital libraries is our best approach for assessing the scalability of HyperDisco.

ACKNOWLEDGMENTS

This research was supported in part by the Danish Natural Science Research Council through Programs 9400911 and 9500996 and the Texas Advanced Research Program under Grant No. 999903-230. The programming work has been carried out during a 7 month sabbatical in 1996 at University of California, Irvine visiting Professor Richard Taylor and the Chimera group. Jim Whitehead and Ken Anderson provided valuable feed-back during demo sessions.

REFERENCES

1. Anderson, K. M., Taylor, R. N., and Whitehead, E. J. Chimera: Hypertext for heterogeneous software environments. In Proceedings of ECHT '94, (Edinburgh, Scotland, Sep. 1994), ACM, pp. 94-107.

2. Bapat, A., Wäsch, J., Aberer, K., and Haake, J. M. HyperStorM: An extensible object-oriented hypermedia engine. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 203-214.

3. Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. The World-Wide Web. Communications of the ACM, 37, 8 (Aug. 1994), 76-82.

4. Bernstein, P. A. Middleware: A model for distributed system services. Communication of the ACM, 39, 2 (Feb. 1996), 86-98.

5. Campbell, B., and Goodman, J. HAM: A general-purpose hypertext abstract machine. In Proceedings of the Hypertext '87 Conference, (Chapel Hill, NC, Nov. 1987), pp. 21-32. Later published in Communication of the ACM, 31,7 (July 1988), 856-861.

6. Dewan, P., Jeffay, K., Smith, J., Stotts, D., and Oliver, W. Early prototypes of the repository for patterned injury data. In Proceedings of Digital Libraries '95, (Austin, Texas, June 1995), pp. 123-130.

7. Durand, D., Haake, A., Hicks, D., and Vitali, F. (eds.). Proceedings of the workshop on versioning in hypertext systems, ECHT '94. Arbeitspapiere der GMD 894, GMD-IPSI, Darmstadt, February 1995.

8. ECHT '92 panel. Open hypermedia architectures and linking protocols. In Proceedings of ECHT '92, (Milan, Italy, Dec. 1992), ACM, pp. 284.

9. Epoch - GNU Emacs for the X-Windowing System. Developed at the University of Illinois, Urbana Champaign. Version 4.2 is available from http://src.doc.ic.ac.-uk/public/packages/epoch/

10. Grønbæk, K., and Trigg, R. H. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 149-160.

11. Grønbæk, K., Hem, J. A., Madsen, O. L., and Sloth, L. Designing Dexter-based cooperative hypermedia systems. Communications of the ACM, 37, 2 (Feb. 1994), 64-74.

12. Haake, A. CoVer: A contextual version server for hypertext applications. In Proceedings of ECHT '92, (Milan, Italy, Dec. 1992), ACM, pp. 43-52.

13. Hall, W, Davis H. C., and Hutchings, G. Rethinking Hypermedia - The Microcosm Approach, Kluwer Academic Publishers, 1996.

14. Hicks, D., Haake, A., Durand, D., and Vitali, F. (eds.). Proceedings of the ECSCW '95 workshop on the role of version control in CSCW applications. GMD-Studien Nr. 289, GMD-IPSI, Darmstadt, 1996.

15. Hypertext '96 panel. Things change: Deal with it! Versioning, cooperative editing and hypertext. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 259.

16. IETF. Uniform Resource Identifier (URI) working group. Further information available from: http://-www.ics.uci.edu/pub/ietf/uri/

17. Instone, K. Report on the hypermedia research and the World-Wide Web workshop, Hypertext '96. ACM SIGLINK Newsletter, 5, 2 (June 1996), 4-5.

18. Kacmar, C. J., and Leggett, J. J. PROXHY: A process-oriented extensible hypertext architecture. ACM Transactions on Information Systems, 9, 4 (Oct. 1991), 399-419.

19. Leggett, J. J. (ed.). Hypertext '93 workshop on hyperbase systems. Technical report TAMU-HRL 93-009, Texas A&M University, November 1993.

20. Leggett, J. J., and Schnase, J. L. Viewing Dexter with open eyes. Communications of the ACM, 37, 2 (Feb. 1994), 76-86.

21. Leggett, J. J., Schnase, J. L., Smith, J. B., and Fox, E. A. (eds.). Final report on the NSF workshop on hyperbase systems. Technical report TAMU-HRL 93-002, Texas A&M University, July 1993.

22. Malcolm, K. C., Poltrock, S. E., and Schuler, D. Industrial strength hypermedia: Requirements for a large engineering enterprise. In Proceedings of Hypertext '91, (San Antonio, Texas, Dec. 1991), ACM, pp. 13-24.

23. Maurer, H. Hyper-G now Hyperwawe. Addison Wesley, 1996.

24. 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, MIT Press, 1989, pp. 107-114.

25. Nürnberg, P. J., Leggett, J. J., Schneider, E. R., and Schnase, J. L. Hypermedia operating systems: A new paradigm for computing. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 194-202.

26. Pearl, A. Sun's link service: A protocol for open linking. In Proceedings of Hypertext '89, (Pittsburgh, PA, Nov. 1989), ACM, pp. 137-146.

27. Rittberger, M., Hammwöhner, R., Aßfalg, R., and Kuhlen, R. A homogeneous interaction platform for navigation and search in and from open hypertext systems. In Proceedings of RIAO '94, (New York, NY, Oct. 1994) pp. 649-663.

28. Rizk, A., and Sauter, L. Multicard: An open hypermedia system. In Proceedings of ECHT '92, (Milan, Italy, Dec. 1992), ACM, pp. 4-10.

29. Schnase, J. L., Leggett, J. J., Metcalfe, T., Morin, N. R., Cunnius, E. L., Turner, J. S., Furuta, R. K., Ellis, L., Pilant, M. S., Ewing, R. E., Hassan, S. W., and Frisse, M. E. The CoLib project: Enabling digital botany for the 21st century. In Proceedings of Digital Libraries '94, (College Station, Texas, June 1994), pp. 108-118.

30. Shackelford, D. E., Smith, J. B., and Smith, F. D. The architecture and implementation of a distributed hypermedia storage system. In Proceedings of Hypertext '93], (Seattle, WA, Nov. 1993, ACM, pp. 1-13.

31. Stallman, R. M. Emacs: The extensible, customizable, self-documenting display editor. In Interactive Programming Environments, McGraw-Hill, 1984, pp. 300-325.

32. Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schütt, H., and Thüring, M. SEPIA: A cooperative hypermedia authoring environment. In Proceedings of ECHT '92, (Milan, Italy, Dec. 1992), ACM, pp. 11-22.

33. Whitehead, E. J. Jr., Fielding, R. T., and Anderson, K. M. Fusing WWW and link server technology: One approach. In Proceedings of the 2nd Workshop on Open Hypermedia Systems - Hypertext '96, pp. 81-86. Available as UCI-ICS Technical Report 96-10, University of California, Irvine, CA 92717-3425.

34. Wiil, U. K. Experiences with HyperBase: A multiuser hypertext database. SIGMOD RECORD, 22, 4 (Dec. 1993), 19-25.

35. Wiil, U. K. Issues in the design of EHTS: A multiuser hypertext system for collaboration. In Proceedings of HICSS-25], (Kauai, HI, Jan. 1992, volume II, IEEE Computer Society Press, pp. 629-639.

36. Wiil, U. K., and Demeyer, S. (eds.). Proceedings of the 2nd workshop on open hypermedia systems, Hypertext '96. Technical report UCI-ICS 96-10, University of California, Irvine, April 1996.

37. Wiil, U. K., and Leggett, J. J. The HyperDisco approach to open hypermedia systems. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 140-148.

38. Wiil, U. K., and Leggett, J. J. Hyperform: A hypermedia system development environment. ACM Transactions on Information Systems, 15, 1 (Jan. 1997).

39. Wiil, U. K., and Østerbye, K. (eds.). Proceedings of the ECHT '94 workshop on open hypermedia systems. Technical report R-94-2038, Aalborg University, October 1994.

40. XEmacs. Developed as a collaborative effort between Lucid, Inc., Sun Microsystems, Inc., the University of Illinois and Amdahl Corporation. Further information available from: http://xemacs.cs.uiuc.edu/

41. Østerbye, K. Structural and cognitive problems in providing version control for hypertext. In Proceedings of ECHT '92, (Milan, Italy, Dec. 1992), ACM, pp. 33-42.

42. Østerbye, K., and Nørmark, K. An interaction engine for rich hypertexts. In Proceedings of ECHT '94, (Edinburgh, Scotland, Sep. 1994), ACM, pp. 167-176.

43. Østerbye, K., and Wiil, U. K. The Flag taxonomy of open hypermedia systems. In Proceedings of Hypertext '96, (Washington, D.C., Mar. 1996), ACM, pp. 129-139.