UNIVERSITY OF SOUTHAMPTON

HYPERMEDIA LINK SERVICE ARCHITECTURES FOR PERVASIVE COMPUTING ENVIRONMENTS

Mark Kenneth Thompson

A thesis submitted for the award of
Doctor of Philosophy

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

April 2005

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

HYPERMEDIA LINK SERVICE ARCHITECTURES FOR PERVASIVE COMPUTING ENVIRONMENTS

by Mark Kenneth Thompson

Recognising the World Wide Web as both a disruptive and pervasive technology, in tandem with the emergence of devices with widely ranging capabilities through which it is navigated, the role of the Web as an information system in pervasive computing is becoming an important concern.

A key characteristic of the Web is the ability to access distributed resources, navigating between them by following links. However, this relatively static interaction model is not suitable in scenarios where resources are not at 'well known' locations, for example when copies or versions exist locally.

This thesis concerns an augmentation of the Web to provide access to spontaneously available local resources, enriched with open hypermedia linking, for local participants in an impromptu network.

The approach taken has been to identify scenarios that serve to scope the application space and then analyse and develop, through a series of prototype experiments, different enabling infrastructures for hypermedia link services, grounded on the notion of a framework of cooperating components.

Contents

List of figures

Acknowledgements

I would like to express my sincerest thanks to my supervisor and mentor Professor David De Roure for his invaluable encouragement and guidance throughout the duration of this PhD, and for the research environment that he has facilitated in which so many new and exciting things happen.

I raise a glass to Hugh Glaser and Tim Chown, for their motivational techniques and counsel for which I am most grateful.

I offer thanks to my colleagues on the Equator project, in particular Mark Weal, Danius Michaelides, David Millard and Richard Beales, for their stimulating discussion and suitable distractions in equal measure.

Lastly, I wish to thank my close friends and family for all their enthusiastic encouragement and support.

– October 2003

For an entertaining and challenging discussion, I would also like to thank my examiners, Professor Kaj Grønbæk and Dr. Les Carr.

– April 2005

Chapter 1 Introduction

Hypermedia has been popularised by the increasing uptake of users to the World Wide Web (Berners-Lee, 1992). The key feature for this is the ability to access distributed documents and navigate between them by following links, i.e. references embedded in the documents currently being viewed.

However, it has been argued that the Web can be seen as nothing more than a glorified file system, much like a hard disk that is distributed across many different storage devices (Nürnberg, 1999). This is contrary to the fundamental concepts envisaged by the early hypertext visionaries such as Bush (Bush, 1945), Engelbart (Engelbart, 1963) and Nelson (Nelson, 1981).

Activities such as the Distributed Link Service (Carr, 1998) and those within the Open Hypermedia Systems Working Group (OHSWG) promote functionality that enables system builders toward realising the goals of the early visionaries. Their enabling idea is the separation of document hyperstructure from the document content, making the *links* between resources first class citizens in the information space.

The Web has become both a disruptive and pervasive technology. It has changed the way that businesses operate and communicate, encouraged millions of children and adults alike to spend free time 'surfing' for content, and provided a valuable tool for teachers to augment and facilitate learning in schools. Its presence at Work, Home and School, is phenomenal, with the advertisement of Web addresses (Uniform Resource Locators, URLs) becoming commonplace on billboards, in newspapers, and on television and radio.

The Web has grown to encompass and deliver many different forms of media, and has become accessible by many diverse access technologies ranging from personal computers to mobile telephones and even the television. However, the infrastructure that delivers Web content, and that which users use to

navigate and retrieve that content, has changed little since the Web's inception. The architecture of the Web, conceptually, follows a straight-forward Client-Server, Query-Response interaction model, relying on a number of underlying Internet standards through well understood access protocols (e.g. HyperText Transfer Protocol, HTTP, and File Transfer Protocol, FTP) atop a uniform TCP/IP network, using a set of common data formats, for example HyperText Mark up Language, HTML, and the Portable Document Format, PDF.

There is a trend away from the traditional 'Alto' (Thacker, 1981) style of interaction when using information systems, such as the Web. No longer is it necessarily the norm to be sat at a desk, using a keyboard and a mouse to navigate and manipulate information. There are a number of divergent trends that could all share the same principles but with different modalities and demands, the most encompassing of which is embodied in a phenomenon labelled as Pervasive, Ubiquitous or Ambient Computing.

It is the advances in network technologies, the acceptance of TCP/IP as a ubiquitous internetworking protocol, and the miniaturisation of computing devices that are providing us with new target architectures, with devices embedded in our environment, in portable or wearable computers, and potentially in everyday artefacts (Estrin, 2000). This results in a particularly profound paradigm shift for the Web, with the massively increased mobility of users, data, services and devices.

The need for an information-oriented infrastructure for pervasive computing is with us now, motivated, for example, by the needs of the mobile business user. From a device technology perspective, we have seen the introduction of a plethora of new computing and communication devices, many of which enable users to browse the World-Wide Web in some form or another (Boriello, 2000). These devices differ from accepted norms for Web access in that they can be aware of the physical context in which they are situated, they can be mobile and they might not be 'always on', they may be sources or sinks of information flows, constantly and in real-time, and they might not be

connected to the one Internet, rather collections of small, impromptu networks.

What does not exist at present is an infrastructure that enables the dynamic manipulation of the interconnections between the resources – the hyperstructure – present in nomadic data across these diverse devices, where the simple, relatively static, interaction model of the Web as discussed above is broken.

The Web can be considered an open system in that the access protocols and resource structure are well documented, and readily adapted. However, it is a closed hypermedia system in that the hyperstructure it affords is rigidly specified, embedded in the mark up of the resources that it serves.

This thesis considers an augmentation of the Web that provides open hypermedia affordances, namely the Distributed Link Service, and investigates different architectures that enable deployment of such services in the new target environments resulting from these observed trends.

1.1 Thesis Structure

The chapter that follows introduces and details a history of hypermedia and its role in information systems, from the early visionaries through the Web and to 'open' hypermedia systems.

Chapter 3 then discusses the field of Ubiquitous and Pervasive Computing, particularly from an information systems perspective. It details a number of projects within that space, and how open hypermedia techniques might provide benefit. It also touches on the inclusion of physical information resources in digital information systems.

Chapter 4 describes a series of scenarios that serve to scope the application space in which information discovery and navigation is desired. In analysing the scenarios, a number of lower-layer systems issues are drawn out and assumptions stated as to the nature of the operating environment for the

information system. Those assumptions made, the latter part of the chapter discusses software infrastructure requirements.

Chapter 5 considers the Distributed Link Service (Carr, 1998) as an appropriate basis for satisfying the requirements of the previous chapter, and discusses its adaptation within a framework of components to provision information discovery and navigation between coincident resources in the context of the scenarios identified earlier.

Chapter 6 documents a series of prototype experiments exploring different infrastructures for the link service component of the proposed framework, the most important component as regards the hyperstructure service provision.

Chapter 7 concludes by reviewing the work undertaken towards the goal of hypermedia service provision in the identified pervasive computing environments, and details areas of on-going and related work in which the author has participated, informed by the findings of this thesis.

1.2 Research Contribution

The following novel contributions result from the work detailed in this thesis:

1. Analysis of the issues regarding spontaneously available local resources as a new application domain for Open Hypermedia systems, in particular, Link Server systems (Sections 4.4, 4.6)

2. Design of a framework of components, and its implementation, grounded on emerging link-local networking technologies to support the discovery and interaction within this new domain (Section 5.2)

3. Examination of a series of eleven prototype experiments that exercise different computing models for an adaptive infrastructure provisioning link services within the framework (Section 6.3)

4. Discussion of the participation of physical artefacts as hypermedia anchors in localised hypermedia systems, enabling their association in hypermedia links (Sections 3.3.2, 4.4.4)

1.3 Declaration

The work in this thesis has been in part undertaken within the EPSRC (Engineering and Physical Science Research Council) funded Interdisciplinary Research Collaboration Equator, grant number GR/N15986/01, and also in part sponsored by an IBM Faculty Partnership Award in co-ordination with IBM Hursley Laboratories, UK. The primary sponsor of this work has been EPSRC through a Studentship Award, number 98319160.

This thesis is based explicitly upon the work of the author within a collaborative research environment. It is all the original work of the author, except as explicitly stated otherwise.

Background discussion regarding Physical-Digital Linking in section 3.3.2 is a result of local collaboration with Drs. Mark Weal, Danius Michaelides and David Millard as part of the on-going Equator project. An early prototype for the LDAP with IRC link service experiment was developed in collaboration with Dr. Danius Michaelides. Aspects of the Ambient Wood and Signage design goals and implementation are the result of collaboration with project partners on the Equator project and within the collaborative environment that is the IAM Group at the University of Southampton.

Chapter 2 Hypertext and Hypermedia

According to the Oxford English Dictionary, hypermedia is defined as

> *A method of structuring information in different media for presentation to a single user, usually through a computing workstation, whereby related items of information are connected in the same way as in hypertext*

Upon looking up the entry for hypertext, the OED quotes the 1965 ACM article by Nelson (Nelson, 1965) in which he coined the term as "a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper" – a definition that we accept as meaning any form of non-linear texts in which the segments, or nodes, of the text are designed to be read in any order as desired by the reader as opposed to the more traditional, linear texts in which that author's intent determines the readers reading activity.

The nodes of information alluded to in Nelson's definition may be fragments of some media, for example a phrase or a paragraph within a document, or they may be a piece of media in its entirety. The resulting information space can be diverse and distributed, populated by multimedia objects that may be persistent, constantly updated, perhaps dynamically generated.

Hypermedia concerns both the form of and the interaction within structured information spaces - information made available by both the presence of, and associations between, media. It has been argued that the associations between information nodes are the key distinguishing feature of such systems (Conklin, 1987).

2.1 Conceptual Pioneers and the Early Systems

Whilst the terminology dates to 1965, the concepts as applied to informational media dates back a further twenty years. The director of the Office of

Scientific Research and Development in the United States, Vannevar Bush, published an article in Atlantic Monthly (Bush, 1945) in which he detailed the design of a number of computing devices that stretched the imagination of a world whose perception of digital computers was strictly limited to number crunching applications, such as ballistic missile trajectories. Most relevant of these for what would turn out to be the hypermedia research field was the Memex machine.

2.1.1 Memex – the Memory Extender

The Memex was a revolutionary, yet imaginary device in which one could store all their books, records, and communications, mechanized so that it "may be consulted with exceeding speed and flexibility". Physically, the design of the Memex resembled a desk with two touch-screen displays and a digital image scanner.

Beyond his remarkable ability to see beyond the limits of technology as was available then, a more notable contribution of his paper was his Information Retrieval thesis. Bush argued that, by mimicking the associative nature of human memory and recall, the machinery of the Memex would be a fundamental tool for succumbing the problems of making real use of information navigation and retrieval in "the growing mountain[s] of research".

By composing sets of associations between digitally captured information, Bush proposed the creation of trails of interest, creating virtual footpaths that could be followed at a later date, annotated or shared with others:

> *When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard. Before him are the two items to be joined, projected onto adjacent viewing positions. At the bottom of each there are a number of blank code spaces, and a pointer is set to indicate one of these on each item. The user taps a single key, and the items are permanently joined* ...

Thereafter, at any time, when one of these items is in view, the other can be instantly recalled merely by tapping a button below the corresponding code space. Moreover, when numerous items have been thus joined together to form a trail, they can be reviewed in turn, rapidly or slowly, by deflecting a lever like that used for turning the pages of a book. It is exactly as though the physical items had been gathered together from widely separated sources and bound together to form a new book.

With the pivotal concepts of the hyperlink and the hypermedia trail conceived, the first implementation was not realised until the early 1960's.

2.1.2 NLS and Augment

Having discovered Bush's article whilst stationed in the Philippines, Douglas Engelbart later joined the Augmentation Research Centre (ARC) at the Stanford Research Institute where he designed a system called H-LAM/T (Human using Language, Artefacts, and Methodology, in which he is Trained) which would "amplify the native intelligence of the user" (Engelbart, 1973). The ideas in the design of H-LAM/T were refined and, with the assistance of his colleagues at the SRI William K. English and John F. Rulifson, Engelbart created the NLS (oN Line System).

Where operators of the period were used to their interaction with computers being solely punched card and ticker-tape, Engelbart's NLS system introduced a radical new interface with television monitors and a novel input device, the bane of occupational therapists in the 21st century – the mouse. Information within NLS was arranged into hierarchies of files, which were in turn hierarchies of statements. Links could exist between any combination of statements and files, providing the first real implementation of a hypertext system.

The system comprised three major components: a database of textual information fragments, customisable view filters that restricted the visibility of information fragments according to both content (based on a high level

analysis language) and depth (simple heuristics for clipping the hierarchies), and views that structured the information being presented.

Yet this was only a small part of what Engelbart's work was about. Besides creating one of the first applications with a full windowing software environment with the mouse-pointing device, Engelbart was concerned with asynchronous, distributed collaboration or teams of workers - essentially Groupware. A version of NLS called Augment was later commercialised by McDonnell Douglas as a system for 'knowledge workers'. His work directly influenced the research at Xerox's Palo Alto Research Centre (PARC), which in turn inspired Steve Jobs and his colleagues towards the creation of Apple Computers. Engelbart and his colleagues were honoured with the ACM Software Systems Award in 1991 for their work on NLS.

2.1.3 Xanadu and Udanax

Whilst Engelbart was working on NLS, Theodore 'Ted' Nelson was developing a unified literary environment whose goal was to be "a universal instantaneous hypertext publishing network" (Nelson, 1988). Named after Samuel Taylor Coleridge's "Kubla Khan", the Xanadu framework was to establish a 'docuverse' in which all information ever published would be stored once and then interconnected with links.

Copyright, royalty payment and (mis-)quoting would be catered for through the development of a linking technique termed 'transclusion', originally penned as 'Xanalogical storage'. With information only ever stored once, virtual copies of document fragments, for example a quotation, are included in-place (transcluded) in the new document, with the original remaining untouched and the two entities inextricably linked throughout all operations. Each time a transclusion takes place, rights management and micropayment can be asserted, and the transcluder assured that a correct representation is taken. Versioning would become a non-issue in that all subsequent versions of a document would be linked from the original, and all versions available for all to navigate.

Xanadu the hypertext system, unfortunately, never materialised as a purchasable product, despite investment by Autodesk Corportation in 1998. Nelson published his design rationale in his self-published books Dream Machines (Nelson, 1976), and Literary Machines (Nelson, 1987). Recently, however, Nelson embraced the emerging trend of Open Source Software development and announced the commencement of the Udanax project, where 'Xanadu Secrets become Udanax Open-Source'.

2.1.4 ZOG and KMS

ZOG was a system developed at Carnegie-Mellon University in the early 1970's, designed to automate task management and on-line reference manuals using a 'general purpose human-computer interface system' (McCracken, 1984). Its success lead to the incorporation of a company called Knowledge Systems in 1981, where work began developing a commercial product, Knowledge Management System (KMS) (Akscyn, 1988).

Both ZOG and KMS were based on a simple paradigm in which information was organised into nodes called 'Frames'. In KMS, each screen would display either one frame or two frames side by side (as in the Memex). Reference and command links were available, indicated by different visual representations. Two link types were available: Tree links indicate hierarchical or structural relationships, for example linking together chapters of a handbook; and annotation links that indicate associative relationships, for example definitions, comments, and cross-references.

Much of the design of ZOG and KMS stemmed from the philosophy that the user interface should be unobtrusive and simple to use. Key examples of this philosophy in use within KMS are:

- a consistent user interface across all operations, including editing, viewing and navigating
- frames are not typed, the data model permits anything to be stored as a frame

- links between frames have no descriptive attributes, it is trivial to follow the link as view its descriptors

2.1.5 Notecards

NoteCards was developed at Xerox PARC by Randall Trigg, Thomas Moran and Frank G. Halasz as an 'idea processing' tool for information analysts. NoteCards was grounded on the concept of paper notecards that are represented electronically as a set of cards stored in a NoteFile. Links were both typed and directional between a source and a destination notecard. Where the source anchor could be a region within a notecard, the destination anchor was always to an entire notecard.

Notecards could be organised by way of 'fileboxes', which were used to categorise or organise collections of notecards. 'Browsers' were specialised cards that contained a graphical representation of a network of notecards, enabling direct visualisation and manipulation of the underlying hyperstructure. An integrated programming environment was provided that allowed NoteCards to be extended to create new applications.

This work was the motivation for Halasz's seminal paper on the critical issues that the community would need to address in ensuing generations of hypertext systems (Halasz, 1988), which is discussed below.

2.1.6 Hypercard

Inspired by the NoteCard activity at Xerox PARC, Hypercard was an application generation utility developed by Apple Computers (Smith, 1988). Hypercard, as its name suggests, uses the card metaphor, where cards are organised into stacks. Hypertext-style navigation is enabled through an event-driven scripting language, where actions are bound to interface elements (for example, buttons) attached to the cards. Anchors may not be placed in full text, and hence must be manually moved if the text changes.

The only built-in navigation aids available are search and history functions. The HyperTalk scripting language not only enabled the creation and navigation of simple links to other cards, but also provided the ability to control sounds, video playback, animation, and, in more recent versions, achieve tighter integration with other operating system controls.

2.1.7 Intermedia

The Intermedia Project (Yankelovich, 1985) was a multimedia environment designed to support teaching in the classroom. Comprising a number of tools, ranging from text editors to 3D model viewers, Intermedia intended to push hypermedia functionality down into 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." (Haan, 1992). This intent was realised in the implementation of a new user shell on top of Apple's Unix implementation, A/UX1.1.

Three major distinctions separated the Intermedia approach from other hypermedia systems of this era. Firstly, nodes ('documents') of the system could be of arbitrary size, not limited to small chunks; Links could be made both from anchor points (termed 'blocks') in a source node, but also to an anchor point *within* a node; and information regarding the hyperstructure (the set of links and blocks, together called the 'web') were stored outside of the documents to which they referred.

Development of Intermedia raised a number of issues that affect link server style hypermedia systems today. For instance, due to the decoupling of hyperstructure from content, the link service had no means of detecting whether a node referenced in a link had been deleted, or its anchor point modified, without performing an exhaustive search of the resource store.

The 'web' prescribed by Intermedia has nothing to do with the World Wide Web as is popular today. It is essentially a unit of context capture that defines a (potentially large) set of related blocks and links, allowing different groups

of users to cluster resources contextually, and share sets of material connected in different ways depending on which web was activated. Upon reflection, the authors realised that the limitation of only being able to apply one web at a time to a collection of documents was unfortunate.

2.2 Issues

In his 1987 survey of the field, Jeff Conklin (Conklin, 1987) devised a loose taxonomy that incorporated many of the hypertext systems that were available at the time and summarised the two core issues that compounded the technologies of the day as the disorientation or 'lost in space' problem (van Dam, 1988), and the 'cognitive overload' problem. The former relates to the difficulty in maintaining a sense of precisely where a reader has got to in the hyperstructure and where they are going. The latter refers to the additional effort required to maintain several trails of thought and paths back through the hyperstructure - an additional load placed on the reader when navigating documents that, in linear documents, is a burden of the author.

Conklin suggests that it is the links that are the defining and essential feature of hypertexts. He also maintains that the other common functionality of these systems, such as windowed views and document processing, are simple extensions from this basic concept. Indeed, a number of more recent projects, notably XLibris (Price, 1998) and Guide (Davies, 1999) are examples of hypertext applications that are such extensions, using small screens to create or access hyperstructure in a familiar manner.

In the survey paper, Conklin identifies the key advantages of hypertext as:

- Reference tracking. *References are equally easy to follow forward to their referent as they are backward to their reference*
- Reference extension. *New references can be made without changing the referenced document*
- Information structuring. *Both hierarchical and non-hierarchical organisations can be imposed on otherwise unstructured information*

- Global views. *Facilities for viewing and modifying the overall structure of complex documents*
- Customised documents. *Threading together the same source text in different ways, allowing the same document to serve multiple purposes*
- Information re-use. *Since the same text segment can be referenced in many places, less duplication of information is required*
- Consistency of information. *As text is moved, references move also*
- Task stacking. *Several paths of enquiry can be open at the same time such that any path can be unwound to the original task*
- Collaboration. *Several authors can collaborate on the editing and annotation of a document*

In 1988, Halasz suggested that there were "Seven Issues for Next Generation of Hypertext" (Halasz, 1988) that the community should address, based both on his personal experience with NoteCards, and with the set of systems that he labelled as a Second Generation system, including Intermedia and KMS. Briefly, these were:

- Search and Query. *Linking alone is insufficient. Effective access to the data is only achieved when navigation-based access is augmented with query-based access*
- Composites. *Groups of nodes and links that may be treated as single, structural entities within the system. A 'head card' might be used to gather a group of associated cards together*
- Virtual Structures. *Structures whose content is determined at the point in time where the structure is accessed, for example, to represent concepts not yet realised*
- Computation. *Move away from the passive nature of existing system to enable internal or external computation engines to either modify existing information and hyperstructure, or generate new*
- Versioning. *Both the complete version history of the hyperstructure as a whole, and that of the individual nodes and links*
- Collaboration. *Support for three classes of collaboration: Substantive activity creating data; Annotative activity creating comments, questions, etc.; and Social interaction, such as discussions etc., arising from the use of the system*

- Tailorability and Extensibility. *Ability to modify the behaviour of the hypermedia system beyond that of a generic information modelling and retrieval tool*

2.3 The Dexter Hypermedia Reference Model

With the hypermedia community steadily producing new systems, often with similar functionality, a need was emerging for some level of interoperation. Existing systems could neither automatically exchange information, nor interact at a systems level. The Dexter Hypermedia Reference Model (Halasz, 1990) was the first attempt at capturing the abstractions found in a range of hypertext systems, with the purpose of providing a mechanism for comparing systems and thus work towards enabling interoperability.

The model divides systems into three layers, as illustrated below. The *Runtime Layer* abstracts the facilities for the presentation and manipulation of the hypermedia network structure.

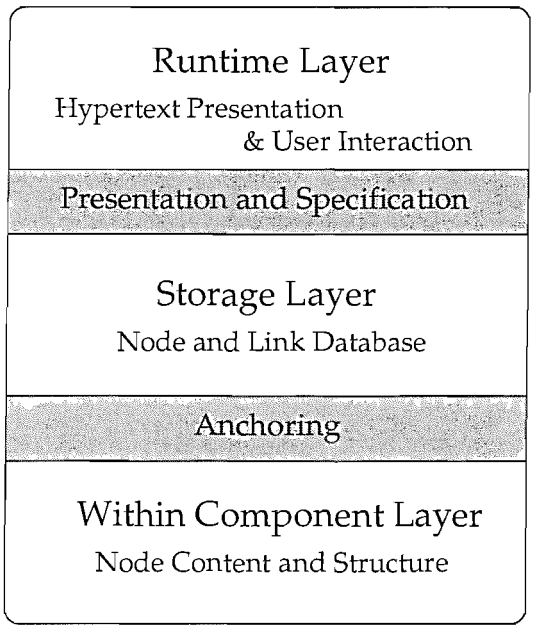

Figure 2-1: Dexter Reference Model

The *Storage Layer* models the basic component (node) and link network structure that is the essence of hypertext, and the core focus of the Dexter model. Links are stored as list of link specifiers, where each link specifier

contains a component identifier, an anchor identifier to determine whereabouts within the (atomic) component the link relates, a link direction and a presentation specification.

The *Within Component Layer* represents the content of a particular component and is, for all intents and purposes, considered to be outside of the Dexter model and as such is largely unspecified.

Between the Runtime and Storage layers, the *Presentation Specification* interface represents a mechanism by which information about how a component that is to be presented to the user can be encoded into the storage layer representation for that component. The way that a component is presented to the user is a function of both the specific runtime layer that is doing the presentation, and also a property of the component or link traversed to get to that component.

Maintaining the opaque-ness of components in the Within Component Layer, the Anchoring interface specifies how links can address the locations of both components and items *within* individual components. The anchors serve as indirect addressing entities that honour the boundary between the hypertext network as modelled in the Storage Layer, and the internal representation of content within the opaque, atomic components.

2.3.1 Issues

Whilst clearly a major contribution to the research community, Dexter was not without shortcomings. Grønbæk and Trigg (Grønbæk, 1992) argued that perfect link integrity hampered link creation and editing, and that there were inconsistencies in the model regarding directionality of links. Leggett *et al.* (Leggett, 1991) discovered that Dexter required significant administrative overhead before being suitable for data interchange between disparate systems. They also note that the model lacked the ability to import partial networks due to the same link integrity issue noted by Grønbæk, and that it could not support separate webs as implemented in Intermedia.

In pursuit of satisfying Halasz' Collaboration issue, Grønbæk *et al.* (Grønbæk, 1993) extended the model to support long term transactions, locking and event notification. Further extensions by Trigg and Grønbæk in 1996 (Grønbæk, 1996) introduced the ability to model both embedded links (World Wide Web-style, see below) and dynamic links (see Microcosm and DLS, below). Even with these extensions, the model was still lacking a complete specification for composite nodes as required by Halasz, nor did it have a notion of a link context (Malcolm, 1991).

The Flag Taxonomy (Østerbye, 1996) models the decomposition of roles within hypermedia systems into interdependent services, enabling researchers to classify their systems in cases where Dexter's layered architecture was inadequate at identifying separate services, such as with Component-Based Open Hypermedia Systems (Nürnberg, 1998). Flag is a conceptual model that decomposes hypermedia systems into functional units and describes them in terms of the *Storage Manager*, the *Data Model Manager*, the *Session Manager*, the *Viewer* and the interfaces between them. Flag was extended in 1998 to encapsulate the interactions between separate OHSs (Østerbye, 1998), and the T3 protocol introduced that enabled application integration such that one application could request that a peer display a particular node, augmenting the traditional application/server integration.

Other models have since been developed that concern aspects of hypermedia research that the Dexter reference model did not sufficiently cover. The Amsterdam Hypermedia Model (AHM) (Hardman, 1994) addressed the issues of node collection, where several nodes are presented together as one, and synchronisation where those nodes are ordered in time as a multimedia presentation. Another, more theoretical, model is the Trellis r-model (Stotts, 1989) in which the notions of hyperstructures are represented as directed graphs of nodes, and connections (links) are extended to Petri nets in which the browsing semantics of the hyperstructure can be specified by attaching semantics to the links between the nodes.

2.4 World Wide Web

By far the most famous distributed hypermedia system in existence is the World Wide Web, a product of the CERN Laboratories in Geneva, Switzerland. Conceived as a technology that would prevent loss of knowledge regarding experiments within the organisation (Berners-Lee, 1989 and 1994), the Web (as it has become known) has since developed into a rich content distribution network pervading the workplace, schools and homes.

Much of the success of the Web can be traced directly to its simplicity. Content servers are distributed throughout the network, whether it be a local intranet or the global Internet, all of which speak a common open protocol, HTTP (HyperText Transfer Protocol), defined by Berners-Lee et al. as an Internet Engineering Task Force standard (Berners-Lee, 1996). Client applications connect to servers to retrieve content, which is then typically rendered in a browser application. The original specifications for the Web had all documents marked up in an SGML-derivate language called HTML (HyperText Mark-up Language). A structured language, HTML could encode presentation structure for documents, for example paragraphs, headings and figures. The hypertextuality of HTML was enabled through the use of an anchor component, which could be used as a reference to another document, perhaps on another server.

As illustrated in Figure 2-2, resources on the Web are addressed by Uniform Resource Locators (URLs). URLs comprise three components: a component denoting the transport and protocol being used for resource transfer (1); an Internet domain name denoting the server on which the resource is located (2); and a protocol-dependent path to the resource (3). There are extensions to this notation that are protocol specific, for example when working with HTTP URLs, the path to the resource can include a fragment identifier that refers to a particular named anchor within the resource.

Note that URLs are not limited to denoting access to a resource by one particular protocol, but also by other open, standard protocols such as *ftp* for file transfer using the FTP protocol, *imap* for mail folder access to messages, and by access to file system local to the client application via *file*. Also, the

path component can also be used as an opaque name or token in protocols where there is no notion of hierarchical organisation. This structured naming approach elegantly provides access to, and unique names for, any addressable resource on the Internet.

URLs can be treated as unique and opaque by applications that do not understand the semantics of its construction, or treated as a location independent token much like a Uniform Resource Name (URN) (Moats, 1997). Where not treated as an opaque or location independent token, it is the job of the client application to parse the protocol used, and then to discover the correct host to connect to in order to retrieve the resource named by the URL. The server then has the duty of parsing any parameterised path, name, query and fragment identifiers to either retrieve the relevant resource from storage, or generate the content for delivery on the fly.

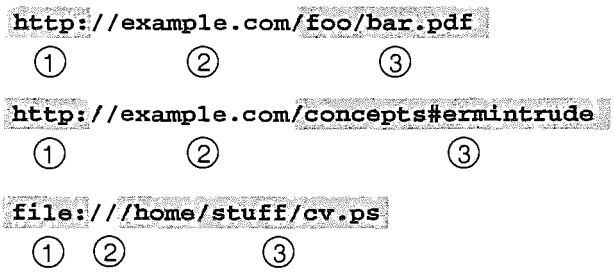

Figure 2-2: URL Components

The first Web browser application did not distinguish between browser or editor roles. Originally called *WorldWideWeb* (Berners-Lee, 1997), later renamed *Nexus* to avoid confusion between the application and the abstract information space, the application was prototyped on the NeXT platform using an application builder suite provided as part of the operating environment. Nexus was essentially a word processor with hypertext functionality introduced by trivial extension of an application object representing text within the document. This extension defined a 'hot spot' or button in a document that, when activated, would result in the focus moved to another part of the document, or a new window being opened with a new document loaded.

As the use of the Web developed, the mark-up became enriched such that more functional documents could be deployed on the web with tables, frames and multimedia objects being the most visually notable extensions. Dynamic content has become a technology of great importance as regards the Web, where documents can be generated by way of a query initiated by the user of the browser, or according to some contextual processing.

The Web Server has evolved beyond being a mechanism for delivering pre-authored, static, documents from a file system or database. Server-side out-of-process scripts using a defined Common Gateway Interface (CGI) offer a programmatic interface for applications to be invoked remotely by clients. User agents such as web browsers or applications can, by passing attribute-value pairs either as the query-part of the URL request or as content data, request dynamic content from the Web Server generated as a result of the query. The advent of CGI and other dynamic content-generating Web Server extensions such as Servlets and Server Modules has led to the Web being used as a platform for application delivery, effectively treating the browser as the GUI, regardless of the host operating system.

The dynamism and adaptability offered by server extensions that allow processes to generate content at request-time, and client-side browser extensions that extend the browser to be an application delivery interface as opposed a document renderer have served to allay claims made about the Web not being a hypermedia system, more a glorified file system (Nürnberg, 1999). Nürnberg and Ashman state that the Web is neither the Open Hypermedia System of the future, nor merely a glorified file-system, rather a synthesis of the two in which lessons from each could be drawn to enhance both fields in unison, for example the simplicity and protocol open-ness of the Web and the separable hyperstructure and rich linking relationships from open hypermedia.

A decade on from the emergence of the Web, the Web browser application has mutated into a generic platform for application delivery. Where early transport was limited to HTML and images, any content type can be delivered over HTTP including audio files, movies, and streamed media (for

which research into the application of hypermedia navigation techniques is ongoing (Page, 2001). The browser has become a monolithic access device, whose complexity in functionality has meant that there are few implementations that offer consistent, uniform access to the diverse resources accessible using the HTTP transport.

Given the mass scientific and commercial interest in developing open standards for new Web technologies, Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology, Laboratory for Computer Science (MIT/LCS) in collaboration with CERN, and with support from DARPA and the European Commission. With the admirable goal of 'leading the Web to its full potential', the W3C serves to promote development of Web standards whilst ensuring interoperability.

Beyond novel extensions to the browser clients and HTTP servers in the original Web architecture, there are a plethora of services that have emerged to enhance the user's experience when navigating the information space. Search engines index web pages and serve to provide access to starting points for navigation on a particular topic, with varying degrees of success. Activities such as the Semantic Web (Berners-Lee, 2001) are investigating techniques that enable such searching to be less variable, utilising metadata mark-up techniques to assist search processes in determining whether a resource would be suitable for inclusion in the result set of a query, modelling the user's intent.

2.4.1 Linking and the Web

As alluded to above, a typical Web link is a point-to-point association between a location in a document and some destination entity, perhaps another document. Anyone on the Internet, or indeed any program or agent (software process), can create Web pages containing such links to an arbitrary object elsewhere.

Two common problems with the Web when held against other hypermedia systems concern a lack of support for hyperstructure, where links are bound

to the resource in which they are embedded, and a simplistic navigation through unary, unidirectional links. Whereas De Young (De Young, 1990) noted that modern (as at 1990) hypermedia systems exhibited complex, typed *n*-ary links allowing more accurate representation of a variety of relationships within a hyperstructure, the Web at inception merely offered embedded, unidirectional linking.

A result of being embedded, links available within Web documents are solely those that the author of the document chose to implement at authoring time. Without the adoption of external mechanisms such as CGI scripts or content-manipulating intermediary services, it is not possible to apply one's own set of links to a resource, nor can the set of links available be adapted at browse-time. This also means that modifying a link requires a change to the document, especially troublesome when the referent is maintained by a third party at whose whim the resource might move.

The fragile nature of these links, where link data appears at every source anchor, is a trade-off between rich semantic functionality and simplicity. It is effective because it facilitates browsing, but it complicates both authoring and maintenance. However, there is nothing inherent in the Web infrastructure that prevents links from being abstracted from the documents and managed separately.

2.5 Towards Open

Hypermedia systems up to the end of the 1980's were largely monolithic applications where all of the functionality for managing content, hyperstructure and presentation were provided by the system as a whole. Indeed, Meyrowitz (Meyrowitz, 1989) notes that the early systems were 'virtually all insular' and that they 'demanded the user disown his or her present computing environment to use the functions of the hypermedia system'. Halasz concluded that these closed systems were no longer viable and that they were to be replaced by 'open' systems consisting of independent components that provide a strict separation between the hypermedia services,

the content, and the applications that are used to interact with them (Halasz, 1991).

Interoperability has long been identified as a requirement for hypermedia systems that intend to go beyond 'toy' status, for example, to be used in large engineering enterprises such as Boeing where hypermedia functionality would need to be integrated into existing toolsets (Malcolm, 1991). Building on the work of projects such as Intermedia, the complete separation of link information from content and the augmentation of third-party applications with hypermedia facilities became axes of interest for a set of activities collectively termed *Open Hypermedia Systems.*

2.5.1 Open Linking

Regarding a link as a *(source, destination)* pair, the location of the link data itself implies the source anchor in embedded-link hypermedia systems. In open hypermedia, the link is a first-class entity that is stored and managed separately ('out of line'), with collections of links bundled into *linkbases*. If a destination changes and this affects just one of these links, just the separated link data requires update rather than all of the source documents that refer to that link. In fact, Open Hypermedia link objects may incorporate multiple sources, destinations and other information.

Having separated the links from the documents, there is now a need for associations between locations in the documents and the individual links in the link database. One approach is a minor variation of the embedded link, where documents contain pointers to the individual links, essentially an indirection mechanism. Looking up the links ('link resolution') can occur before the documents are delivered, perhaps by a web site authoring tool, or during user interaction. The latter is akin to clicking on a link that submits a query to a search engine.

However, Open Hypermedia advocates exactly the opposite: the separated links themselves contain sufficient source anchor information to identify the source locations in the documents; for example, a byte offset, coordinates or

an index to a particular word. The tremendous advantage of this model is that the source material can now remain in its native format. For example, it could be a multimedia format that does not support embedded links. We can also now apply arbitrary sets of links to read-only materials, and alternative sets of links to single sets of documents.

The separation of resource from its relation to other resources in a hyperstructure not only enables multiple sources and destinations for links that result in a richer set of relationships between documents, but it also enables systems to present a different hyperstructure around a document (different sets of links to and from) depending on the user's preference or context – a technique often labelled as Adaptive Hypermedia (Brusilovsky, 1996, De Bra 1999). An example of this approach is in educational systems where advanced students are presented with a different hyperstructure to those that are less advanced.

Inferences can be drawn from the nature of the association between nodes linked together, indeed a number of taxonomies have been proposed that describe various link types in hypermedia. Ashman (Ashman, 1997) classified links into *Hand-made* and *Computed*. Computed links were categorised as either being *Pre-computed*, where the relationship is generated and stored persistently for future use in a linkbase, and *Dynamic*, created only when the need arises by result of, for example, a query. Should a dynamic link be stored for future reference, it becomes a *Cached link*.

Beyond the nature of link generation, Lowe and Hall describe a taxonomy of the relationships between the links and the objects to which they link that falls short of specifying full semantics, but does provide a useful mechanism to describe the relationships from a high level (Lowe, 1999). They state that if the relationship is one aimed at the organisation of an information space, the links reflecting this relationship are 'structural'. If the relationship relates to the content of the information space, then the links embodying this relationship are 'associative' or 'referential'.

Associative links embody a relationship between two concepts that are independent while referential links convey a connection between a concept and more details or further explanation for it. Further discussions of link typing and associated taxonomies can be found in (De Rose, 1989; Allan, 1996).

2.5.2 Open Hypermedia Systems

Davis *et al.* (Davis, 1992) define a hypermedia system as being open provided that:

1. The hypertext link service was available across the entire range of applications available on the desktop. Where applications would not be aware of, or know how to manipulate anchor identifiers in the Dexter sense, it would be necessary for the system to hold links *and* their anchors separate from node content

2. The link service must work across a network on heterogeneous platforms, implying that functionality is provided by a number of communication processes. This defines the notion of a link service as being a framework for routing messages regarding hypermedia system interactions between various components

3. The architecture should be such that the functionality of the system can be extended. This implies that a modular design with a well-defined programming interface is implemented

4. There should be no artificial distinction between author and reader, rather that is a function of the application using the open system, or even the operating system on which the applications are running.

Subsequently, Davis (Davis, 1995) summarises the ensuing debate regarding the definition of open hypermedia, stating that a truly open hypermedia system should be open with regard to:

1. Size. *Nodes, links, anchors and other hypermedia objects should be permitted to be added to a system without limitation*
2. Data Formats. *Any data format, including temporal media, should be permitted*
3. Applications. *Any application should be allowed to access the link service in order to participate in the hypermedia functionality*
4. Data Models. *New models may be incorporated such that external hypermedia systems may be interoperated with, and data exchanged between them*
5. Platforms. *It should be possible to implement the system on multiple distributed platforms*
6. Users. *Support for multiple users, each with their own view of the objects in the system.*

Whilst Davis notes that no one system succeeds when assessed against all of the above criteria, systems exist that conform sufficiently that they warrant the term 'open'. A representative sample of these is presented in the sections that follow.

Pearl (Pearl, 1989) also offered a concise summary of the open approach stating that 'an open system implies that the system can be extended to mediate different heterogeneous implementations; integration implies much greater cooperation and conformity'.

2.5.3 Sun's Link Service

Sun's Link Service, developed at Sun Microsystems by Pearl (Pearl, 1989), saw the first practical implementation of hypermedia interoperability through the definition of a link service. Part of their commercial development environment, NSE, the link service was composed of a protocol specification, a link server program together with a library that could be used to turn any application into a client of the service. Links were stored and managed separately from nodes that they associated, and were combined at run-time as applications requested the node.

Several text editors, a file browser and a project-scheduling tool were shipped as 'link aware', together with various third-party CASE tools and utilities for managing the link databases. By utilising the programming interface of the library provided, applications could be developed or ported to also benefit from the hypermedia approach.

The service as deployed only provided un-typed links with no descriptive attributes, such as 'direction'. Different media could be incorporated with ease for the generation and display of anchors was outside the remit of the link service. However, the link functionality of the link server itself could not be extended.

Despite Pearl's desire to 'see linking, and attendant hypertext capabilities, as much a standard part of the computer desktop as the cutting and pasting of text are today', Sun's Link Service failed to penetrate the market. Whether this failure was due to a reliance on third-party vendors adapting their own applications or due to a lack of a comprehensive linking model, work on Sun's Link Service did highlight a number of important issues for the development of open hypermedia systems:

1. User interface. *The reliance on third-party editors to handle documents hampers any drive for consistent user interfaces*
2. Document control. *Unless documents reside in a document management system, they can be modified, deleted or moved without the hypermedia system's notice*
3. Link consistency. *If a document is modified, deleted or moved, any links pertaining that document may become inconsistent*
4. Distribution. *Where monolithic systems function within the context of one file system, an open hypermedia system will likely involve resources on other, possibly remote, file systems*
5. Collaboration. *It is natural for there to be a requirement of multi-access on resources held within a hypermedia system for the purposes of collaboration. This raises access and editing issues including sharing, locking and permissions*

6. Versioning. *Both the documents and the hyperstructure that relates them may be versioned, raising issues as to which particular version a link refers to, or what a hypermedia system should do if a subsequent version of a document does not contain the anchor to which a pre-existing link referred to in a previous version.*

Pearl also offered a concise summary of the open approach stating that 'an open system implies that the system can be extended to mediate different heterogeneous implementations; integration implies much greater cooperation and conformity'. With regards the Link Service, Pearl stressed that simplicity and lack of user restriction are key characteristics of protocols between open systems components in order to preserve autonomy of the individual tools whilst providing some level of integration.

2.5.4 Multicard

Multicard (Rizk, 1992) was the result of the Espirit project Multiworks, providing a toolkit that enabled programmers to create and manipulate distributed hypermedia structures. Components of the system would communicate requests using an open and extensible protocol called M2000. Components, such as Document editors, were not required to implement the full protocol: a minimum of 'Open/Close Node' was deemed sufficient in order to access each document from a distributed storage database.

The toolkit defined a range of classes for managing hypermedia objects such as nodes, groups (composites), anchors and links. Links were implemented as communication channels between their associated objects. A variety of messages were defined, with associated behaviours attached, for example a link activation message would represent the action of traversing the link and opening the associate object. Nodes, groups and anchors could have event-driven scripts attached to them that defined system behaviour in an extensible way dependent on the messages received.

Multicard separated the notion of author and viewer by requiring nodes and groups be created using a separate utility, the hypermedia authoring tool.

Once the structure existed, any M2000-compliant editor could be used to navigate the hyperstructure and edit node contents, or script event actions.

Links were pair-wise associations in the Multicard model, treated explicitly as communication channels between the objects that they associate. The authors argued that the event-based communication model for links with scriptable behaviours for link endpoints allowed Multicard to be readily configurable and therefore available for a greater range of applications than existing hypermedia toolkits.

2.5.5 PROXHY

Another link server system, PROXHY (PROtotype implementation Of an eXtensible HYpertext system) (Kacmar, 1991), was developed as a modular, object-oriented system. It provided a simple hypermedia data model in which anchor objects would connect application objects (nodes) to link objects, where the anchor and link objects that defined the hyperstructure were maintained separately from node content.

The architecture defined for the system consisted of four layers, which could comprise of a number of processes: The Back-end Layer defined a storage interface to a persistent data store for the hyperstructure components, for example in a database or on a file system; The Hypertext Layer comprised the Anchor and Link objects that were implemented as a set of communicating processes; The Communications Layer comprised a series of message routing objects that were responsible for routing messages between components; and the Application Layer defined the integration with third party applications.

Objects within the application layer were required to implement at least a core set of messages pertaining to the management and presentation of anchors, and conform to PROXHY's unique naming convention for objects.

A product of its modular design, PROXHY exhibited significant flexibility. Should a particular implementation of an anchor not support a message type, the message router processes would dynamically traverse the inheritance tree

and despatch the message to a parent object, enabling 'lazy' extension of functionality for programmers. Also, should an application not be modifiable to interoperate with PROXHY's messaging scheme, their integration can be achieved through the use of *proxy* anchors. Proxy anchors capture enough information about the applications to which they relate such that when activated (e.g. by a link containing the anchor as a destination being traversed), the system can recreate the desired application context.

What PROXHY offers by flexibility and extensibility in both design and implementation, it lacks in speed, the overhead of indirect message passing between objects proving too great to attain satisfactory performance.

2.5.6 Hyper-G

Hyper-G was an open hypermedia system developed at the Graz University of Technology. A large-scale, distributed, multi-user, hypermedia information system, its principal focus was on providing structured access to the management and use of University information. In particular, four key areas:

- Research. *Enabling instant access to publications, experimentation results, digital libraries*
- Teaching. *Supporting teaching and learning, delivering structured course content*
- Administration. *Managing the business process of University administration, minutes, rules, and records*
- Communication. *Enable collaboration and communication between groups of individuals across each of the above domains.*

Users submitted documents encoded as HTML (see 2.4 above) to Hyper-G's storage manager, an object-oriented persistent database. Documents could then be grouped into aggregate collections, which in turn may belong to other collections. Hyper-G supported a notion of metadata that included keywords for resource searching and automatic link generation using utilities provided.

As with other open hypermedia systems, hyperstructure was maintained separately from content, yet unlike other systems Hyper-G did not relax the link integrity mandate of the Dexter model, rather it would ensure that no links were presented to clients of the system that associated non-existent nodes. The linking model provided was restricted, with only bi-directional binary links supported, although link anchors could be aggregate collections. There was no notion of contextual webs, or link databases, preventing authors from reusing resources in new contexts.

Being designed as a multi-user system from the outset, Hyper-G provided registered users with their own private information spaces in which they could collect their own resources. All objects within the system conform to a user-based permission system, similar to the Unix user model, thus allowing collections, documents and links to be secured in a consistent manner.

There were two principal client applications providing access to the Hyper-G service: *Harmony* on Unix, and *Amadeus* on the Microsoft Windows platform. World Wide Web browser clients such as Netscape Navigator could be used to access the information space, although without the extended features provided, such as the hyperstructure map visualisation and contextual collection browser as found in Harmony that collectively reduce the 'lost in space' problem often perceived with large-scale hypermedia networks.

Hyper-G was later released commercially as Hyperwave (Maurer, 1995).

2.5.7 DeVise Hypermedia (DHM)

Based firmly on the Dexter reference model, DeVise Hypermedia (DHM) was an object-oriented application framework by Grønbæk and colleagues at the University of Århus in Denmark (Grønbæk, 1994). The DeVise project had the aim of producing a set of general tools to support system development and collaborative design in application areas involving 'hypermedia in the large'. An example of the target scope of their application is engineering design projects, such as those envisaged by Malcolm (Malcolm, 1991).

In developing DHM, a number of areas of the reference model were highlighted as being inadequate with regards the requirements of modern, open hypermedia systems, leading to a number of extensions to the model as discussed above.

Firstly, DHM allowed 'dangling links', or links where one or more endpoint nodes did not exist in the storage layer. The authors point out that this is a necessary and natural extension of the model in that it would then permit systems to be developed with an asynchronous garbage collector, where processes could, at a later date, correct un-resolvable or incomplete links through deletion or manipulation. When nodes were deleted from the system, it would then not require all anchors and therefore links that reference those nodes to be deleted also. This is a familiar scenario to embedded-link type systems such as the World Wide Web, where ownership of the data in which reference is made of a node may result in the anchors or links not being accessible for such updates.

DHM also revised Dexter's notion of link directionality. Whilst specifying that links have direction, Dexter did not specify what semantics such a term would be associated with. The authors identified three notions of direction: *Semantic*, *Creation*, and *Traversal*. The initial implementation avoided the issue by creating all links as bi-directional at creation-time, with users having the ability to edit this should they wish. Further refinement lead to DHM supporting links with more than two endpoints, and the authors have demonstrated DHM, and thus an extended Dexter, supporting embedded links and Microcosm-style generic links in an integrated, object-oriented manner (Grønbæk, 1996).

Collaboration support was achieved in DHM through consideration of role-based ownership of data, requiring node and structure locking, and through support for structure versioning (Grønbæk, 1993). The authors subsequently developed technology for deploying these enhanced-Dexter services in a World Wide Web environment (Grønbæk, 1997).

2.5.8 Chimera

Chimera (Anderson, 1994), developed at the University of California, Irvine, applied open hypermedia techniques to the process of software development. Chimera allowed developers to associate different objects in the system regardless of type, and visualise those relationships within their software development environment.

The data model employed resembled Dexter, with the removal of the link integrity constraint, and the addition that anchors could associate multiple views of the same object. The authors state that the concept of multiple views on the same node could be neither modelled nor specified in Dexter. Links could be also be *n*-ary, associating collections of anchors, for example a definition of a software component, its documentation and an example of it in use.

Chimera consisted of a central server that maintained the storage and delivery of the hypermedia objects, termed 'hyperweb', based on a set of hypermedia events that clients declared interest in. Applications would communicate with the server using remote procedure calls to interact with the hyperstructure.

The onus of presentation of objects, views and anchors was strictly on the client applications, often resulting in an inconsistent user interaction style. The fact that presentation was devolved to the client also meant that each client was responsible for any persistent storage required for viewer-specific data.

The Chimera team tested their theory on application integration by integrating the Adobe FrameMaker desktop publishing suite. Wrapper processes managed the task of translating FrameMaker's data model and hypertext concepts to Chimera's, and custom application macros were written such that the appropriate Chimera messages were interpreted and generated as necessary. Later work with Chimera saw experimentation extending Chimera's functionality as a link service for the World Wide Web (Anderson, 1997).

2.5.9 Microcosm

Microcosm (Fountain, 1990) was developed at the University of Southampton by a team of researchers with a need to be able to provide a hypertext environment for the presentation of existing archival data (video disks, sound and image libraries as well as textual documents). The aim was to use those data sets for scholarly enquiry and teaching.

In dealing with archival material, it was a requirement that hyperstructure was separate from data structure because it was not possible to embed anchors inside the read-only materials. Further strict requirements were self-imposed by the team in that they desired a loosely coupled component-based system that could be flexible in both implementation and configuration, and that there should be strictly no distinction between the role of author and user of data in the system.

2.5.9.1 Architecture

Microcosm did not employ a client-server architecture, but instead distributed the various hypertext functions into a chain of processes which communicated by message passing (Heath, 1992). This enabled the behaviour of the system to be modified easily at run time, adding different link database processes or swapping history management and tours facilities in and out depending on the system behaviour required by the user, which in turn may be a function of their technical or subject sophistication.

Document viewers and editors were instances of processes that fed and sourced from the chain, and similarly communicated with other system components by message passing.

The two core components comprised a Document Control System (DCS) that managed the interaction and launch of document viewing processes, and the Filter Management System (FMS) that would control the chain of 'filter' processes that provided the hypermedia functionality in Microcosm. The filters would process messages that would add, alter, or delete links from the set applicable to a particular resource being viewed. This modular approach permitted straightforward integration of new functionality to the system,

such as user browsing history tracking and map visualisations (Wilkins, 1994).

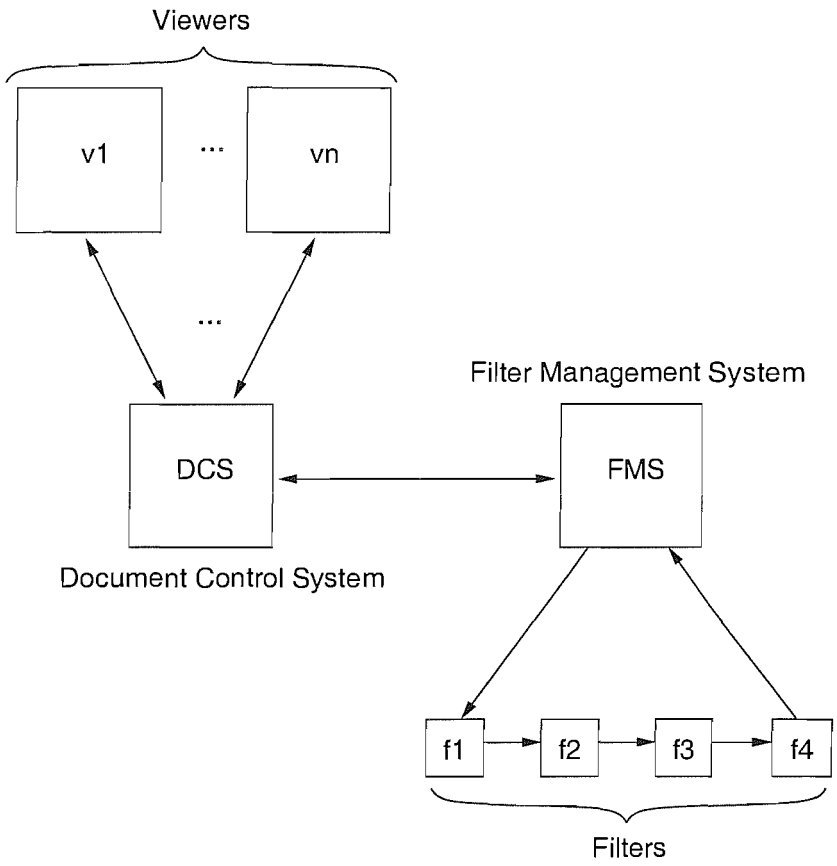

Figure 2-3: Microcosm System Architecture

2.5.9.2 Application Integration

As regards open design and observations made of other open systems, Microcosm identified different levels of third-party application integration that would enable hypermedia functionality to become commonplace on user's desktops:

- Fully aware. *Applications either tailor made or ported from source to incorporate the full feature set of messages and interactions*
- Partially aware. *Where complete integration was not possible, application features such as macro interfaces could be scripted, or application proxies written, to provide partial integration, e.g. Microsoft Word*

- Unaware. *Applications that cannot be adapted to enable Microcosm functionality. Rather, more contrived and potentially less reliable solutions involving, for example, the paste-buffer were required.*

With an emphasis on integration with third party applications, it was inevitable that some desirable functionality could not be ubiquitous across all viewers, such as anchor highlighting. Instead, a separate 'available links' window could be displayed containing a list of links available at a point in time explicitly queried by the user.

A novel technique called the Universal Viewer (UV) provided a subset of the full Microcosm hypermedia functionality to unaware or otherwise closed applications, for example the Mosaic Web browser. The UV application parasitically 're-parented' the target application within its operating environment, and interacted with it and the Microcosm services through provider- and user-defined macros that performed keystrokes or application menu actions (Knight, 1995).

2.5.9.3 Microcosm Linking

The Microcosm team realised that open hypermedia could be taken a stage further, relaxing the association between source anchors and specific locations in specific documents, so that one source anchor can match many different locations to which it applied (Davis, 1992). This was achieved through content-based techniques, and a link using one of these flexible source anchors was termed a *generic link* (Fountain, 1990). In the case of a text document, the source anchor might be a word, so that every occurrence of that word results in the link being available, a technique often utilised as a glossary facility. In other media, the source anchor might identify a feature to be matched; for example, a link from a particular kind of image, or a fragment of a song. Also, given that the database of links was being maintained by sets of filter processes, it was possible to compute links dynamically, making associations at request-time based on the known set of anchors available.

Hall (Hall, 1994) desired to 'End the Tyranny of the Button' by urging that users be encouraged to expect to query the system for links in much the same

way as they might query an encyclopaedia for information. By offering dynamic and computed links such as generic links, every object in the document potentially became the source of a link. In Microcosm, the interface to hypertext functionality was achieved through a user explicitly marking a selection (e.g. by highlighting a word or phrase with the mouse) and then having the system identify whether the selection, or any part of the selection, represented a resolvable anchor. This made it possible for users to ask what links are available within, for example, a whole paragraph of text and for the system to dynamically compute what link were available at query-time rather than relying on pre-authored buttons.

2.5.9.4 Issues

The Microcosm architecture meant that it was difficult to present a consistent and coherent interface to the user, with the usability of the system suffering as a consequence. Work by Weal on SHEP (Screen Handler Enabling Process) (Hall, 1997) provided a central controlling mechanism that could arbitrate between external screen management processes and the Microcosm interface components.

The nature of the Microcosm model, with sequential chains of components whose messages are arbitrated by centralised manager processes, also lead to communication and start-up inefficiencies. Later work on Direct Communication (Wilkins, 1994) and Microcosm TNG (Goose, 1997) have adapted the model to incorporate different communication strategies and distribution of services within the architecture, with extensions that would support multiple users (Hill, 1994).

Microcosm was later commercialised, and later extended to incorporate the Web as a delivery platform in a product called WebCosm.

2.5.10 Open Hypermedia Protocol (OHP)

With the emergence of numerous open hypermedia systems, such as those discussed above, one might envisage a utopia of intercommunication and interoperability. However, this was not the case. In an attempt to remedy this

situation, the community of researchers investigating open hypermedia began a series of workshops, the first being held at the European Conference on HyperText in Edinburgh, 1994 (Wiil, 1994). At their second workshop, they formed a working group (OHSWG) as an instrument to address the important issues of standardisation and interoperability.

The OHSWG began to define a protocol, called the Open Hypermedia Protocol (OHP) (Davis, 1996), as a common language that would enable communication and interoperation between all open hypermedia systems. Hypermedia servers would export functionality to hypermedia clients, using protocol 'shims' as translators where necessary, thus reducing hypermedia application development times, enabling greater interaction with other diverse systems, and promoting re-use of existing tools and code across different hypermedia platforms.

Initial designs for OHP were criticised as being inconsistent within itself (Anderson, 1997a), for being overly complex and large (Millard, 2000), and for lacking an underlying data model and architecture (Anderson, 1998).

In an attempt to make the problem space more manageable, the working group divided OHP into sub-domains. OHP as was became OHP Navigational Interface (OHP-Nav) and its functionality reduced so that it focused on navigational hypermedia exclusively. Other domains were identified for Taxonomic and Spatial hypermedia, and also for functions otherwise unconsidered, such as unified storage interfaces, OHP-Store.

Work by Millard *et al.* has begun to address the data model issue, inspired by the inter-domain OHP activity, with their proposed Fundamental Open Hypermedia Model (FOHM) (Millard, 2000a) and associated link server application, Auld Linky (Michaelides, 2001).

2.5.11 Distributed Link Service (DLS)

The Distributed Link Service (DLS), an activity of the University of Southampton, was a hypermedia linking service designed for use in

conjunction with World Wide Web resource servers. One of the goals of the DLS was to improve the overall connectivity of documents on the Web, but with the benefits of Open Hypermedia techniques and other experiences with link services gained by their work on the Microcosm project (Carr 1995 and Carr, 1996).

The model has given rise to additional user- and application-level interactions with linkbases and documents, such as link integrity checking where link services ensure that the resources the link points to is still valid before offering it back (Davis, 1998), and dynamic linkbase generation through keyword extraction (De Roure, 1998). The DLS afforded an unenclosed environment that provided an additional navigational overlay to otherwise 'closed' Web pages, where linking is either missing or inadequate.

2.5.11.1 Architecture

The term 'distributed' in the name indicated that the link data was maintained separate from document content, and that the link services could be remote services across different applications, beyond SunLink and Microcosm that augmented applications local to one machine. This arrangement was designed to provide a powerful framework to aid navigation between sets of documents across different applications, enabling ease of link authoring, in order to solve some of the issues of distributed information management (De Roure, 1996).

The DLS was a further move away from monolithic hypermedia systems to a point where hypermedia linking could be afforded to all applications that could interact with the Web for information retrieval.

The early implementations of the DLS server were sets of CGI scripts on off-the-shelf World Wide Web servers that offered functionality similar to the Linkbase filter in Microcosm. The server would communicate with a program integrated with the host application to which hypermedia functionality was being extended, much like the Universal Viewer in Microcosm, as discussed above (Davis, 1994). A core difference between the approach used with the DLS and Microcosm's UV was that the DLS interactions took place as HTTP

requests over a TCP/IP transport, the same protocols used by the rest of the Web, whereas a custom protocol was employed by Microcosm.

The resolution of the links occurred at the point at which the user expressed an interest in a particular word or paragraph of the document. The service offered functionality that allowed the user to create links interactively or to view different sets of links from different link service providers. In the initial case, the application whose hypermedia functionality was extended was the Netscape Navigator Web browser application. Clients could query the set of available links from the current viewed document based on the contents of the entire select and words or word-pairs within the selection. Link authoring was enabled using a Web form interface, invoked either manually or as a result of a client action on the application. The full range of Microcosm-style links were available, including generic links, offering a simple mechanism for document connectivity.

Link creation and resolution were services that may be provided by a number of link resolution engines. The DLS used resolvers that matched against terms or combinations of terms within the documents being served, queried against static linkbases to determine the terms' potential as link anchors. These resolvers were then hardwired into one monolithic service, spawned as child sub-processes, or chained sequentially (De Roure, 1999) such that each one saw the document with links added by the previous resolver as it passed along the chain, c.f. Microcosm's Filter chains.

The different software modules that could inject various kinds of links to the documents enabled a user-specific navigational overlay that could be used to superimpose a coherent interface to sets of unlinked, or insular resources, such as archives of otherwise non-marked-up media. An example of this technique used with an otherwise unlinked resource set is the Open Journals project (Hitchcock, 1998).

The application re-parenting approach developed for Microcosm's Universal Viewer prove problematic for the developers of the DLS. Each update to the underlying application being wrapped with DLS functionality required more

work to be done to re-integrate the client side 'shim' that provided the interface to the DLS server. This problem was accentuated with the release of Microsoft Windows 95 (the previous versions were implemented in Windows 3.1), which saw a major overhaul of the operating system internals that made the shim approach unviable.

To combat this problem, the developers moved to an intermediary model (Barrett, 1999), where the DLS added links and annotations into documents as they were delivered through a proxy from the original WWW server to the client browser.

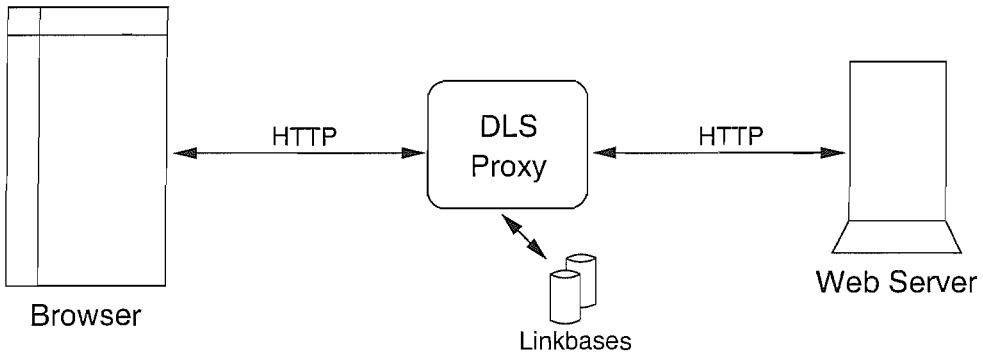

Figure 2-4: DLS Proxy Architecture

This architecture is conceptually simple and served to make the service transparent to its users by embedding it in the World Wide Web's document transport system. However the implementation created a number of new issues. Depending on the desired characteristics of the application in which the DLS was employed, different modes of operation were required that varied the moment at which link resolution occurred, for example before or after the delivery of a document.

If before, links that might have been available would not be known by the client, or presented as unresolved buttons; if after, the user could adopt a more query-oriented mode of interaction, especially with generic linking. This is an inherently synchronous arrangement that delayed the eventual delivery of the document as the amount of processing expended in determining links increased with each additional process placed in the path between content server and client.

With both deployments of the DLS, the configuration of the service, for example which linkbases were available and what particular style of links were to be provided, was controlled by a Web delivered form-based interface. Different linkbases could be turned on or off for a given session of use, whether explicitly by the user, or automatically by the system on computation of applicable contexts.

The ability to apply different sets of links to the same set of resources gives rise to a weak notion of context, similar to that seen with the Intermedia project. A stronger notion of context, where automated document processing employing information retrieval techniques, such as Term Frequency-Inverse Document Frequency (TF-IDF), has been developed in the QuIC project (Queries In Context) such that different linkbases could be applied automatically, determined by the computed context in which the resource query was made (El-Beltagy, 2001).

Historical implementations employed a tight coupling between the proxy component that analyses the information stream from the upstream content server (or chained proxy that might itself be another link service) and the link resolution engine. Decoupling the proxy component from the link resolution engine permits more flexibility, for example, the ability to plug different resolution back-ends on, perhaps at run-time, permitting a much more flexible customisation than previously possible.

2.5.11.2 Distribution

In (De Roure, 1996) the functional components of the DLS were considered for distribution such that link resolution did not take effect in one relatively heavyweight server (the DLS). The architecture that emerged in prototypes resulting from that work, shown below, proposed that DLS components could distribute link resolution messages to multiple link service components, aggregating the results. The authors identified the trade-off between increased scalability, localised fault tolerance, and greater coverage over

increased latency and long component dependency chains increasing the complexity of fault tracing.

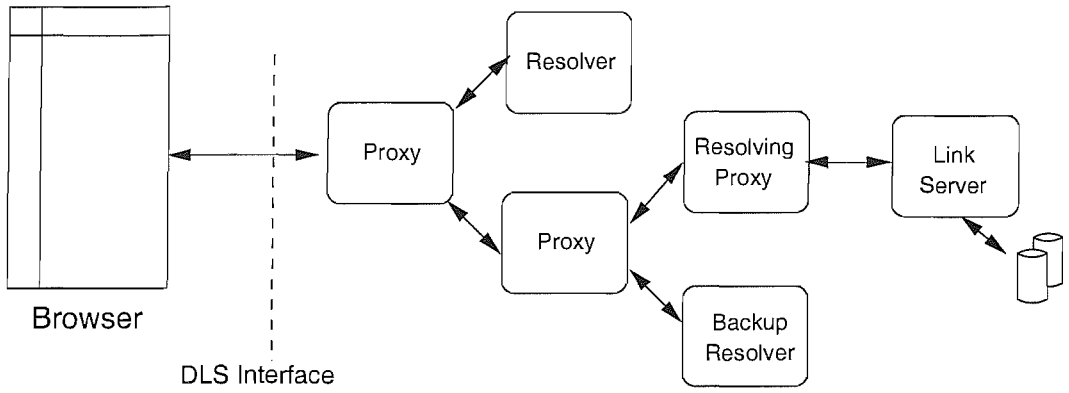

Figure 2-5: Distributed DLS, adapted from (De Roure, 1996)

This enables a different distribution of the distributed link service: Where the early DLS separated the relationship between link data from the resources over which it identifies structure, this decoupled DLS facilitates the distribution of the processes of link resolution from the information path between client and resource server.

2.5.11.3 Alternative query architectures

Research by De Roure and Walker (De Roure, 2000) began to examine alternative technologies to HTTP-based transport for the DLS. The notion of query routing, *"I don't have a link, but I know someone who might"*, was examined using standard distributed directory service technologies including Whois++ (Deutsch, 1995) and LDAP (Yeong, 1995).

As part of their early experiments with query routing for link services, De Roure and Walker identified the emergence of different linkbase domains, and offered a set of categories:

- Links associated with a given user
- Links associated with a given subject area
- Links associated with a given administrative domain, e.g. a department
- Links supporting a specific task, e.g. teaching
- Links associated with a given group of users; e.g. shared bookmarks

- Links maintained by a given publisher of information
- Links generated as a result of specialised search algorithms, such as feature matching

Their work was the start of the move away from intranet link services, such as those deployed on local and enterprise networks, with an aim to provision customised information spaces to improve the effectiveness of interaction with information in systems such as the Web on larger scale or in less pre-determined networks.

Activity within the Web's standards body, the World Wide Web Consortium (W3C) has seen the emergence of a number of linking standards that, whilst awaiting complete implementations, bring open hypermedia linking to the Web in a manner that might benefit link server systems such as the DLS. XLink, the proposed standard for hypertext linking (De Rose, 2001), allows databases of links to be maintained separately from the documents, with each of the links is fixed to specific regions of specific documents. Whilst its defining working group declared XLink a recommendation (therefore 'standard') mid-2001, the uptake of the technology has not been as widespread as many expected. This is perhaps in part due to the tolerance that users and developers demonstrate for the shortcomings of HTML in-line linking.

2.6 Seven Issues (revisited)

A side effect of the Web's success with regards the Hypermedia research community is that there is no longer the drive to develop systems that are not Web-aware, and that interoperability with the Web, at least on some level, is cited in hypermedia research papers. At the 2002 Hypertext conference, Halasz's issues papers were revisited and several new 'core issues' introduced (Whitehead, 2002), reflecting the current trend of hypermedia research and its interest in World Wide Web technologies. The issues, in rank order as voted by the conference delegates, were:

1. Economics. *Assessment of the cost of applying and using hypermedia technology on the Web*
2. The Evil Click. *Correction of the uni-directional, untyped, replacement-action link as seen in Web-based hypermedia*
3. Physical Linking. *Introducing physical artefacts as hypermedia anchors*
4. Alternative models of expression. *Ending the tyranny of the* `<a href=>`
5. Personalisation. *Adapting both availability and presentation of information to the context of the user*
6. Pervasive Hypermedia. *Integrate hypermedia technology with everyone's infrastructure such that anything and everything on the screen can be linked*
7. Archiving. *Beyond versioning, introduce system support for historical archiving of information.*

Grønbæk's issue of Physical Linking, and an extension of Anderson's call for pervasiveness of hypermedia on the desktop to the environment are reflected as motivations for the work of this thesis.

2.6.1 "404" and problems with Open-ness

One of the most common grievances aired by users of the World-Wide Web is the error page served up by the site when following a link in a document to a destination that no longer exists, the '404' error page. This may happen for a variety of reasons, the most common being that the document has been deleted or moved, resulting in a 'dangling link' in the source document.

This may not be an issue in a personal, or even enterprise, site where content management systems can be employed to maintain the integrity of the associations between the documents they control, but it is a severe problem when the source of the link association is not within the sphere of influence of the site designer, for example in a user's bookmark collection. Hyper-G (Andrews, 1995) provided a solution for the problem of broken links in the World Wide Web's open environment by restricting link editing to guarantee consistency and integrity.

A similar issue is present in other open hypermedia systems. Maintaining data integrity when anchors that reference data items are widely distributed without any inverse knowledge as to where those referents are stored is hazardous. Where one system or user owns the entire data set of nodes being referenced, the problem may be scalable, for example by scripting tools that compute integrity across all anchors and all links that associate those anchors. However, when the anchors or the links are owned by a third party, some higher-layer protocol action is required.

This issue could be overcome by insisting that either the System or Author inform any link service with associations referencing the resource when it has been moved or deleted, or offered a destination refinement when a document is edited. Davis (Davis, 1995) suggests that link integrity is a resolution-time problem, and that within the description of the link, two new timestamp tags should be added for the source and destination document. Upon resolution, a process can be invoked by the link server to confirm that the associations are not older than the document. If this is the case, that is the document has changed, an integrity checker can be employed to ensure that the links are still valid and inform the user that the document has changed. Davis refers to this as the 'editing problem'. Note that closed systems that use embedded mark-up to specify anchor locations do not suffer from the editing problem, for when the content moves so too do the anchors.

Inconsistent linking can result where third party applications have different levels of engagement with the open hypermedia system. The position within a resource to which an anchor refers may not be navigable from the OHS given the interface to the application. Taking Microcosm as an example, a fully aware viewer would have no problem navigating to an exact byte offset within a document as an anchor point for a link. A partially aware viewer might be able to navigate the interface caret to the nearest paragraph block, and an unaware viewer may only be able to open the resource, thus rendering the link as if it were a whole-resource link, not a specific target within that resource.

2.7 Summary

The three identified fathers of modern hypertext concepts each have different end-goals for their ideas. Where Bush intended for hypertext systems to model the human mind and thus make them familiar and easy to use, Engelbart chose to augment the user's abilities where tasks would be impossible without the technology. Nelson desired ubiquitous access to and interaction with all information everywhere.

One of the commonalities of these three visions is that hypermedia should assist the user in navigating the available information space. This navigation may involve pre-authored links ('buttons') and links computed dynamically (Hall, 1994). The former is very much the interaction style realised by the World-Wide Web, where the links are embedded in the source document; the latter is a more query-oriented modality.

Whilst not an exhaustive survey, this chapter has introduced and detailed a history of hypermedia and its role in information systems. Popularised by the World Wide Web, several efforts have adopted open hypermedia techniques and applied them to the Web to enhance the discovery, navigation and delivery of information.

The motivation for the work reported in this thesis is desire to extend the added value of hypermedia-enriched information spaces from their current domain of pre-determined, pre-configured local and enterprise scale networks into domains where the arrangement of information, services and users are less organised.

We suggest that an appropriate and thus far successful technique for enriching the linked-ness of the Web is the notion of *open linking*, of which the DLS has been shown to provide a straightforward, elegant and scalable solution in existing enterprise-scale applications.

The next chapter introduces the domain of ubiquitous and pervasive computing, and suggests how hypermedia techniques might be applied in such environments where the monolithic single-network architecture of the

Web struggle to deliver information discovery, navigation and retrieval affordances that would otherwise be welcomed.

Chapter 3 Ubiquitous and Pervasive

This chapter introduces the field of Ubiquitous and Pervasive Computing, and suggests how hypermedia techniques might enable access to information both 'digital' in the familiar sense of documents and annotations, and 'physical' in the form of posters, signs and artefacts.

3.1 Pervasive Computing

Pervasive Computing is a cross-disciplinary area extending the application of computing to diverse usage models. The preface to a new conference series, the International Conference on Pervasive Computing, notes that the field encompasses a broad set of research topics such as low power, integrated technologies, embedded systems, mobile devices, wireless and mobile networking, middleware, applications, user interfaces, security, and privacy (Mattern, 2002).

The pervasive computing vision was established by work at Xerox PARC that started in 1988 on 'ubiquitous computing' (Want, 1995 and Weiser, 1999). Weiser is credited as fathering the notion of ubiquitous computing, a philosophy based on the premise that the most profound technologies are those that disappear, ones that weave themselves into the fabric of everyday life until they are indistinguishable from it (Weiser, 1991).

Weiser observed that the disjunction between the Real World and the world of the computer had nothing to do with the tasks for which people were using computers, but that the problem was more fundamental. He and his colleagues believed that the idea of a 'personal' computer was misplaced, and that trends towards miniaturisation into laptops and personal digital assistants were merely transitional steps towards achieving the real potential of information technology. Rather, their vision sought a world where computing devices would be periphery to the focus of their users, demanding less concentration and awareness that traditional, 'Alto'-style, workstations.

The goal was for casual interaction to result in rich information with "all the advantages of an intelligently orchestrated and highly connected computer system" (Want, 1995).

Weiser explains in (Weiser, 1993) that their idea of ubiquitous computing first arose from contemplating the place of the computer in actual activities of everyday life. In particular, anthropological studies of work life demonstrated that people primarily work in a world of shared situations and unexamined technological skills whereas the computer was isolated and isolating from the overall situation, and fails to get out of the way of the work. In other words, rather than being a tool through which we work and so which disappears from our awareness, the computer too often remained (and still does remain) the focus of attention.

This desired world of abundant, un-tethered, portable, even wearable devices is made possible only by a unique combination of two powerful trends: Moore's Law (Moore, 1979) and the emergence of high speed wireless communication networks.

Moore's Law postulates that the number of transistors on a given chip can be doubled every 18 to 24 months as a result of the continued shrinking in the size of constituent components. This suggests an increase in computational power per unit surface area of silicon. An alternative perspective is that the same level of functionality can be achieved in half of the space, driving towards miniature devices and pervasive computing.

As Want observed, computing technology was already (in 1995) becoming more and more prevalent in common appliances such as home audio-video equipment, kitchen appliances and portable digital assistants, but without interconnection.

Ark and Selker (Ark, 1999) introduced an issue of the IBM Systems journal with an essay that placed the shift to pervasive computing in context with the developments in computerisation in the 1990s, and the resulting impact on the way humans interact with machines. They stake their claim that the

human-computer interaction community had already been working towards 'disappeared' computer interaction without branding the research topic with a name. They also explicitly acknowledged the fact that the terms *pervasive computing* and *ubiquitous computing* are synonymous, an observation echoed by the Editor-in-Chief of the IEEE Pervasive publication series (Satyanarayanan, 2002). Satyanarayanan also comments that the other 'visions' that followed Weiser's, such as *proactive* and *autonomic* computing overlap with ubiquitous computing, but are not strictly subsets of the same research.

3.1.1 ParcTab

Early research projects, particularly at Xerox PARC and EuroPARC, considered the notion of communicating contextual information to be paramount in ubiquitous computing systems. This was the ability of components of the system to share information about their status, the user and the environment or context in which they were operating. This context information might include such elements as the name of the user's current location; the identities of the user and of other people nearby; the identities and status of the nearby coffee machines, and other devices; and also physical environmental parameters such as time, temperature, light level and weather conditions (Want, 1995).

Much of the work was very forward looking, often having to design for technology that was not likely to be commercially realisable for ten years or more in the future. Characteristics such as electrical and computational power requirements are often measured against Moore's law, and projects in the ubiquitous computing space subsequently trade-off the construction of an operational research system against something that resembles an optimal design that might be realisable in the future.

In the case of the early ParcTab work at Xerox, devices were designed on three scales of size: a 'tab' device that fits in the pocket; a 'pad' that would fit in a wallet or briefcase; and a 'board' that would, for example, be mounted on a communal wall. The devices and the software architecture that supported

them were developed to study various issues among which were: the design of user interfaces for small devices; the design of location-aware applications; and the use of thin-client systems to handle mobile users.

Four application axes were chosen as being representative of the set of applications typical to work-related technology:

- Information access applications. *Calendar applications, Both access to and automated contextual tours of information present 'in a room'*
- Communication. *Readily available, contextually-aware, email-style communication for conferencing and paging*
- Computer Supported Collaboration. *Support for co-operative activities such as annotation and voting*
- Remote Control. *Controlling applications and appliances that typically take their input from a keyboard, mouse or switch*

The applications were not restricted to the control and interaction with other software applications, but also the environment in which the devices were situated, for example with the Responsive Environment Project (Elrod, 1993), the Tab, Pad and Board devices enabled automation and control of room lighting and heating.

Aside from the technical obstacles of implementing technology that would not be feasibly realisable for another decade, Weiser's evaluation of their early work highlights some interesting factors affecting acceptance that were largely aesthetic. For example, the physical size and appearance of the devices; the convenience of the devices and the peer pressure of the evaluators insisting that they were used; and the limited application types deployed.

3.1.2 Aware information

In a review of the field of Ubiquitous Computing with regards mobile systems, Scholtz (Scholtz, 2001) describes work at the Defence Advanced Research Projects Agency (DARPA) in the United States that explored the

notion of 'aware information'. Scholtz makes the observation that in many application scenarios in the ubiquitous, mobile, proactive, wearable and context-aware research spaces, the common goal is not to provide *computation* on the move, but to provide *information* that is aware of the user's situation and able to deliver itself appropriately.

Scholtz summarises four sub areas of the collective research fields that are of importance in ubiquitous computing research:

1. Implicit Interaction. *If the system has an awareness of the situation, the environment and the aims of the user, it would be able to reduce the need for explicit human-computer interaction (Dey, 2000 and Schmidt, 2000)*

2. Task-based Interaction. *The ability for a 'task' to follow the user and automatically take advantage of whatever computing resources the user's current environment has to offer*

3. Nomadic Information Management. *The ability for information to be accessible regardless of location, possibly involving automatic replication*

4. Adaptable Software Architectures. *The need for a common execution environment that all devices can support such that the above requirements can be met.*

Scholtz also notes that the conclusion to Weiser's 1993 ACM article in which he states that ubiquitous computing "is likely to provide a framework for interesting and productive work for many more years or decades, but we have much to learn about the details" does not need updating, that many of the issues first drawn out are still not fully understood.

3.1.3 Current Status

In their 2002 article (Want, 2002), Want *et al.* review the progress of the community as regards development of hardware towards Weiser's vision. They note that the four most notable improvements in hardware technology

in the decade that followed the early ubiquitous computing work have been the development of wireless networking, processing capability, storage capacity and high-quality displays.

They also observed that, due in part to the success of the Web and the subsequent ubiquity of Internet access, consumer demand for 'pervasive technology' in the shape of personal digital assistants and multi-purpose mobile telephones indicate that the market is ready for more advanced new technology, and that this adoption requires common standards across many products and locales.

The vision for these devices may stem from work such as Weiser's, but there are examples of such pervasive devices much earlier on in computing history. As discussed in the previous chapter, Vannevar Bush is most famous for the conceptual Memex device that served as an anchor point in history for browseable knowledge. The very same article also contained descriptions of other devices rarely cited, but are in-keeping with the notion of pervasive computing. These include context capturing devices such as the Cyclops Camera that "worn on forehead, would photograph anything you see and want to record. Film would be developed at once by dry photography"; and a 'vocoder', "a machine which could type when talked to" (Bush, 1945)

Pervasive computing is a vast research area that touches on many disciplines including hardware design, software architectures, human-computer interaction, networking, and communications research. Excellent surveys of the field can be found in (Hansmann, 2001) and the inaugural edition of IEEE Pervasive magazine, edited by Satyanarayanan (Satyanarayanan, 2002), in particular, the article on the progress of hardware development towards Weiser's vision by Want *et al.* (Want, 2002) and on systems software architectures by Kindberg and Fox (Kindberg, 2002a).

3.2 Pervasive Information

While new devices and networking are the subject of much research and development, together with new interfaces and applications (Abowd, 2000),

there are also many research issues relating to Weiser's 'fabric' itself. Not only has there been an explosion in the number of networkable tiny devices, the amount, richness and diversity of digital content has observed a similar growth.

Techniques for dealing with adaptive and scalable delivery of multimedia content are attracting much attention. Existing ideas include the request-time transformation, or 'transcoding', of data between media formats and the controlled degradation of, for example, images to a different resolution – all dependent on the capabilities of the requesting device and the bandwidth restraints of the network, typically using intermediary (proxy) technology (Fox, 1996; Smith, 1999; Bickmore, 1999; Barrett, 1999).

Some content is delivered in a store-and-forward manner, some is streamed either from stores or live. Devices can capture events in a space and communicate them, or store them for later perusal. Information can be augmented by annotation. The information around us is both rich and dynamic, and it forms a complex distributed system in its own right. Internet and Web technologies have emerged to address some of these issues. The extent to which we can borrow ideas and technologies from the global information infrastructure and apply them in this pervasive setting is of interest to this thesis.

Research assessing the impact of a limited display area on user comprehension of documents has suggested that there is no severe restriction imposed by a small display, such as that on Personal Digital Asssitants like Palm Pilots or iPaq hand-held computers (Dillon, 1990). However, the artefacts imposed by a hypermedia system, such as the icons to invoke link generation or resource publication, may have a stronger impact than the survey suggests.

Some systems, such as Carmeli's Personal Information Everywhere (Carmeli, 2000), deal with the small input area and lower-power processing capabilities of hand-held devices by treating them merely as lightweight thin clients,

where applications and data are all held elsewhere on the network and the hand-held is just a window onto an application that is running elsewhere.

Activities such as PowerBrowser (Buyukkokten, 2000) and Digestor (Bickmore, 1999) sidestep the issue by re-purposing Web content for different target devices, for example, by resizing images or restructuring textual content by thematic analysis and contraction.

What follows are some example projects that are looking specifically at aspects regarding information delivery in pervasive computing environments, chosen for their consideration of a hypermedia or hypermedia-like approach to information interaction.

3.2.1 CoolTown

Birnbaum introduced a Hewlett Packard view of Pervasive Computing in (Birnbaum, 1997) where he quotes Allan Kay of Apple Computer Inc. as pointing out that "only people born before a technology is invented think of it as technology". Drawing parallels to the usability of television sets, he states that information technology needs to transcend merely being able to be manufactured and commonplace, but become intuitively accessible to ordinary people, delivering sufficient value to justify the large investment needed in the supporting infrastructure.

Alluding somewhat to the emergence of the CoolTown project (Barton, 2001), Birnbaum noted that the invention of the World Wide Web browser had provided a 'giant step' toward the first requirement, and that the Internet phenomenon had addressed the second.

The CoolTown project is an on-going research effort at Hewlett Packard that looks to extend the World Wide Web to the physical world, combining web technology, embedded web servers, and wireless communication to develop systems supporting nomadic users.

The project considers the real-world Web different from the conventional World Wide Web in that links can be discovered by sensing the physical world in addition to browsing Web pages. URIs for services and content are sensed through technologies such as infrared 'beacons', barcodes, RFID tags, and iButtons, by the devices carried around by nomadic users.

These URIs define the 'web presence' of the corresponding entity, and offer a mechanism to manipulate artefacts in the real world just as they would documents on the Web. The identifiers are bound to physical or virtual resources, of which there may be many, and are interpreted by resolver services either automatically (implicitly) or explicitly through user choice, based on the application at hand (Kindberg, 2002).

By engaging the same open standards as are used on the Web, CoolTown avoids many of the issues of adaptable software architecture as summarised by Scholtz. Entities in CoolTown are 'Web-present', building on adaptable, open standard Web technologies, rather than present on some platform independent interface such as Java or CORBA. Kindberg states that this is due to their belief that content-oriented computing, as opposed to object-oriented computing, led to the Web being successful, and that the same rationale applies in nomadic computing (Kindberg, 2001).

The CoolTown team acknowledge that it is unclear whether the Web paradigm is the correct one for pervasive computing, whether the cognitive load it presents and its efficacy for the desired activities are adequate. They take a quantum leap in assuming that people can accept the different types of physical hyperlinks (barcodes, etc.) as readily as they have the 'underlined' link on a Web page. They also fail to address the impact of 'dangling links', links to resolvers or resources that are either not present or inconsistent with the identifiers to which they are bound.

3.2.2 GUIDE and GeoNotes

Projects that consider information provision cued on physical location typically concern museum, tourist, or business interests. The notion of

situated context, placing users or artefacts in locations and using that placement as a primary data source, has been evident in a number of information systems in pervasive computing environments, particularly GUIDE (Davies, 1999), CyberGuide (Abowd,1997), and Hippie (Oppermann, 1999).

We consider two here, one formal visitor adaptive information system, GUIDE, and one social annotation-based system, GeoNotes.

3.2.2.1 GUIDE

GUIDE was a joint project between the Computing Department at Lancaster University and Lancaster City Council to investigate the provision of context-sensitive mobile multimedia computing support for city visitors. The goal was to develop a number of hand-portable multimedia end-systems that would provide information to visitors as they navigated a city in which wireless networking was prevalent, incorporating systems that were context-sensitive. The notion of context employed by GUIDE included knowledge of users, their environment including, most importantly, their physical location.

The GUIDE infrastructure comprised high-bandwidth wireless networks (cells) spread throughout the city of Lancaster across which mobile clients would download information to guide users as they navigate the city, supporting interactive, highly dynamic information services such as tourist guides, ticket booking and intra-cell messaging. Each cell would be served by its own server, with local storage and processing providing tailored information to the end-systems within that cell.

Having observed that the information model present in existing GIS and contextual ubiquitous computing information systems were insufficient for their requirements, GUIDE designed their own model comprising objects for physical locations of interest and objects for navigational way-points, with attributes such as weight metrics for distance/means of transport.

Additionally, the model incorporated information structured in a hypertext in that objects could contain references to documents, thus providing users and

applications with multiple entry points into the hypertext information base. The model also offered a level of dynamic adaptability in that documents within the information base could be modified at request-time, for example based on number of visits to location/current status of attraction.

The overall GUIDE system can be viewed as a central web server that mobile clients access via the wireless network. In order to improve the performance of the system, each cell of the system has its own cache such that requests are, where possible, satisfied locally. In addition, GUIDE utilises a pre-emptive caching technique by which a subset of the contents of each cell's cache are sent using IP multicast to all users within the cell (i.e. all users on the same wireless network). This has been shown to improve response times, scalability of the wireless network and battery life of the mobile devices (Davies, 1999).

3.2.2.2 *GeoNotes*

GeoNotes is a project at the Swedish Institute of Computer Science (SICS) focusing on providing 'social' virtual annotations for the physical world. In GeoNotes, this information is in the form of "virtual Post-It notes that can be read by other users passing by the physical location where you placed the note" (Espinoza, 2001).

Annotations are stored as data items in a central spatial database, with a server component that manages the interactions between distributed clients and the database backend. Metadata regarding annotation use-patterns is also maintained and referenced when calculating appropriateness metrics, offering a level of collaborative annotation filtering to the service.

In the current implementation, clients are Java applications using remote method invocation (RMI) to interact with the server as part of a services framework also developed at SICS, sView. sView (Bylund, 2001), a platform for 'personal services', offers an execution environment for services with mobility and persistence characteristics.

There are three modalities of use for the GeoNotes client, reflecting the three anticipated levels of engagement with the system. Firstly, a geographically constrained active search mode offers an explicit means by which annotations can be sought out, pulled, from the annotation server. A mixed push-pull interaction exists enabling serendipitous navigation of the available annotations, where an overview of available annotations cued to situated context is made available (pushed) for users to browse and then retrieve individual entities (pull). Finally, a notification-based interaction mode exists where annotations are pushed to the client as and when they become contextually relevant.

GeoNotes strives for socially enhanced digital space in the sense of less formally defined information cued to physical location, such as restroom graffiti or office sticky-notes, arguing that a 'fun' and expressive system would be more dynamic and subversive than other location-based systems that serve commissioned, formal, content. It would be a technically trivial extension beyond the annotation of physical locations to explore more adaptive information. For example, 'social' documents comprising transcluded annotations (due to Nelson), filtered by some user model and with the application of hyperlinks between locations, annotations or other resources. This could offer different perspectives on the physical space in which the user is situated with somewhat more cohesion than the "Check this out! It's cool!" type of annotations reported in the GeoNotes literature.

3.2.3 Geo-Spatial Hypermedia

The Disappearing Computer Initiative was a 2001 EU initiative to investigate methods and techniques for embedding computing in everyday artefacts and the resultant interactions that are possible. Projects within the programme range from service-oriented frameworks for the management of intrusive notifications (Jonsson, 2001) through to the enrichment of physical paper to make it an effective resource for interaction with electronic media (Frohlich, 2002).

The collective aim of the sixteen projects within the programme is to define new concepts and techniques upon which applications can be enabled in which the computer has 'disappeared' into the environment. Specifically, the programme description states the focus of research as three-interlinked objectives:

- Create information artefacts based on new software and hardware architectures that are integrated into everyday objects

- Look at how collections of artefacts can act together, so as to produce new behaviour and new functionality

- Investigate the new approaches for designing for collections of artefacts in everyday settings, and how to ensure that people's experience in these new environments is coherent and engaging

As part of their Disappearing Computer Initiative project *WorkSPACE*, researchers at the Centre for Pervasive Computing at the University of Århus have experimented with extending the abstract notion of Spatial Hypermedia, where visual means are employed to structure information within *n*-dimensional virtual space, to physical space. This work has included the integration of techniques from spatial hypermedia, geographical information systems, location-based services and collaborative virtual environments within a prototype of an information organisation tool, Topos (Grønbæk, 2002).

Topos concerns Workspaces that are sets of spatially related and placed materials such as documents, 3D models and annotations. These workspaces can be composed by linking, where the spatial context of the user is replaced upon link traversal to the new workspace, or by composition, where workspace proxys manage the interaction between the two information contexts. The notion here is that resources for a particular topic or activity can be collected into spatial workplaces, and then inter-related with geo-spatial cues.

The work reported in (Grønbæk, 2002) was geared towards a scenario of collaborative work support for professional landscape architects, where information pertaining projects is in workspaces distributed between people on-site and back at the office, for example to assist with contextualised decision making and problem solving.

This project is pervasive computing in the sense that it addresses the distribution of information to nomadic devices in the field and senses the physical situation of participants using the system as part of the information context.

From a systems perspective, the Topos system employs a selective replication model for the distribution of content between participant nomadic devices, where revisions to the data are (explicitly) synchronised when appropriate. It also takes an event-based publication-subscription approach to distribution of collaboration messages such as changes to the workspace and object database, employing a compression technique such that as little information is communicated between participants as is acceptable to maintain a usable system.

The current focus of the project is specifically the integration of spatial hypermedia to GIS applications and as such the outstanding issues reported concern the relationships between spatial hypermedia and physical spatial information management. Whilst the work reported mentions annotations anchored on physical location, similar to GeoNotes above, there is no mention of how information that is external to the system might be integrated, for example, documents (plans, drawings, annotations) that a contractor has on-site on a laptop but not integrated to the geo-spatial hypermedia system.

3.3 Relevance of Hypermedia

The nature of content in information-rich pervasive computing applications is often pre-authored and somewhat static, especially when concerning applications such as museum guides and tourist information applications.

Dynamism and adaptivity of the presented information has been introduced cued on sensed contexts such as physical location, user task or environmental conditions. In systems such as GUIDE, content is delivered as per a template in which dynamism is achieved through the run-time substitution of variables within those templates. For example, delivering seasonal opening times according to the current date, and advertising different activities according to the current weather.

Content may also be adapted according to user preferences and experiences, perhaps offering additional technical information regarding museum artefacts if a user has indicated that they are an expert or have specific interests. Information may also be dynamic in the sense that it is not strictly authored by an authority for a particular purpose, but opportunistically by a community such as with the physical location annotations of GeoNotes.

In the previous chapter, we observed that the majority of hypermedia systems that exist are intranet based with a traditional client-server model based on large servers accessed by many clients and interconnected by a static network, the largest example being the World Wide Web. In an Internet-scale Web context, much of the same observations apply regarding the nature of the content: it is largely static, with pre-authored structure offering navigation between resources on a large-scale network.

The very nature of pervasive computing, its nomadic and embedded devices, its interconnectedness and ubiquitous nature, not only suggests that computing devices are everywhere, but also that information is everywhere.

Where research such as Adaptive Hypermedia and Open Hypermedia have added dynamism in both selective presentation of information, and dynamic navigation between information resources on the Web, little work exists that tests the applicability of those techniques within pervasive computing environments. We suggest that the techniques of open hypermedia provide a means of structuring this rich information space.

Activities such as WAP (Wireless Application Protocol)[1] and PIE (Carmeli, 2000) are part of the infrastructure for mobile and pervasive computing environments, providing some level of support for scenarios where devices other than just laptops are mobile with access to the Internet for published data, but they do little to change the static model.

WAP compliant devices are typically clients, and content creation is confined to short messages and audio transmission. The WAP protocol suite provides a mechanism for mobile phone and Personal Digital Assistant (PDA) access to a scaled-down 'version' of the Web with its transport (WTLS) and mark up (WML and WMLS) standards, corollaries of HTTP and HTML/JavaScript that provide the same functionality in the 'Alto' Web-world, with a similar set of affordances and issues.

These approaches and others like them are somewhat limited in that they all assume connectivity to a wide-area network, and they do not address the hypertext issues concerning navigating (linking) between resources that coincide in and around the user device's locality.

3.3.1 Generic Linking

An example of an applicable and beneficial hypermedia technique includes generic linking. Generic links support a world where hyperstructures persist but content is dynamic in that they enable new content to be linked up on the fly. They are therefore suited to the pervasive world of dynamic content, especially where pervasive devices are also used for content capture – newly generated content can have generic source anchors applied, and therefore be participative in the hyperstructure in relations to other resources.

Generic links promote link reuse and are a powerful aid to a more explicit form of link creation using a 'sculptural' approach (Weal, 2001): by invoking appropriate sets of generic links, e.g. specific to a subject domain, many alternative links are identified throughout a document. These can then be

[1] Open Mobile Alliance: http://www.wapforum.org/

filtered, for example by process or by manual author selection using some sculptural tool, with the appropriate generic links or their particular instantiations stored in a new linkbase for publication.

3.3.2 Physical-Digital Linking

First he runs through an encyclopaedia, finds an interesting but sketchy article, leaves it projected. Next, in a history, he finds another pertinent item, and ties the two together. Thus he goes, building a trail of many items. – Vannevar Bush, As We May Think (Bush, 1945).

Links in Bush's scenario represent relationships between nodes, in this case microfiche documents, which today would be Web documents in browsers. However, this definition of 'linking' can be broadened to include links between any objects, be they in the digital world or the physical.

The Web browser has become the ubiquitous interface to information systems, with the links appearing explicitly in the interface and providing the means of interaction. In a pervasive setting, we suggest that the browser is neither the primary nor only interface. The user should interact directly with their environment, be it physical or virtual. For example, their actions may form a query that results in information being displayed in appropriate modes on appropriate devices. The intention is that hypermedia services run behind the scenes ('in the fabric') to support this interaction.

Without a browser, links can still be used explicitly for interaction, but through other interfaces such as augmented reality (Sinclair, 2001) or some other form of integrative technology between the physical environment in which the user is situated and the digital world of the resources they are interacting with. These resources may be local, within their context (e.g. enterprise) or as 'far away' as the Internet.

As demonstrated in many of the systems discussed in the previous chapter, a link is comprised of a number of parts. For the purposes of clarifying the

terminology used in the rest of this document, the following definitions can be taken.

3.3.2.1 *What Constitutes a Link?*

A link can be deconstructed into three key components. The link object itself, the anchors that the link object connects, and the actual objects – be they physical or digital – that the anchors reference. Taking these in reverse order:

Objects

These are the 'things' that are to be associated. In an ideal environment, there should be no restrictions on what these 'things' are. They could be a person, a document (physical or digital), or an entire city. More often than not, these objects are resources, instances of 'things'. However, the thing being associated may be conceptual, incurring a reification or resolution process when being navigated.

For example, if there existed a need to create a link to a person, what is the nature of the thing that would be linked to? The physical entity somehow digitised and 'caught' up in the hypermedia web? Perhaps the role of the person, for example 'Head of School' is captured, or the intention could be that the link is to serve as a mechanism for contacting the person, i.e. the result of following the link is to end up with a communication channel directly to them. Alternatively, and perhaps most commonly used, it is the *concept* representation is the one that is anchored upon.

Anchors

A link connects two concepts, with each concept representing an object, possible a real world artefact or perhaps – as is the case with traditional hypermedia systems – a digital artefact. This is analogous to the 'Sign' of Saussure (de Saussure, 1971). The anchor serves as a *signifier* or *Sound Image*, something that allows the actual object to be discovered or traversed to, and a *signified* or *concept*, the actual object.

Digital anchors are well understood and quite straightforward. Taking HTML as an example, an anchor is an explicit element of a document, embedded within its mark-up. The HTML model is confusing in that links are partially grounded: the association is explicitly anchored on a particular element at its source, and the location of the link target (a complete resource, or another anchor within a resource) is specified at source as an attribute of the anchor.

Considering physical anchors that participate in cross-medium associations, or *Physical-Digital links*, the work by Kindberg (Kindberg, 2003) offers devices such as bar codes as physical anchors that can be transmogrified into digital anchors by resolution to a URI through scanning and directory look-up, and research by Want (Want, 1999) and more recently by Grønbæk (Grønbæk, 2003) uses Radio-Frequency Identifier (RFID) technology to do likewise.

Resolution of Anchors

In what we would term traditional hypermedia links, the anchor contains information that specifies the source object, often in explicit terms, for example a URL that describes precisely where the object resides. The resolution process of the anchor often involves little more than retrieving a document from a file system by means of a filename. This is still a resolution process but the process and the reference are often viewed as the same thing.

When links can have physical anchors, their anchor resolution can become much more complex. Take for instance a link that has a painting in a gallery as its source. The link anchor will could contain a symbolic representation of the painting, for instance its name "Bathers at Asnières by Georges Suerat". If someone is standing in front of this painting in the National Gallery in London, how does this get resolved as the link anchor specified? There are many different strategies for this, for example:

- The physical location of the person in the gallery could be compared to the physical location retrieved from a database mapping symbolic names to physical locations.

- A barcode could be placed next to the painting, which when scanned, resolves a digital anchor for the link.

- An image analysis system identifies the image detected by a camera carried by the visitor as the painting having the symbolic name given by the source of the link.

Links

The final important component is the link object itself. The link holds references to any participant anchors and also contains additional information such as a description of the link, and perhaps other metadata information regarding features including the directionality of the association, information as to the provenance or source of the link, etc.

The notion of a physical anchor can be extended to include more contextual information, as opposed to simply tagging a 'thing' to allow it to partake in some hyperstructure. For example, the physical token that represents an anchor in a hyperstructure could, with sufficient capture technology available, represent an association between the collection of people in a meeting, the time and place that the meeting took place, and the current agenda item at the point where the anchor was created (e.g. the barcode printed or the RFID tag detected and therefore bound to the captured context). This moves beyond a barcode or RFID tag serving as a physical identifier for a single physical document toward the notion of a physical token that represents the link between multiple objects, both digital and physical; a digi-physi-link.

Alternatively, given a contextual hypermedia model such as FOHM (Millard, 2000a), the physical token might represent the set of context keys that scope the applicability of different association structures in a linkbase. In this approach, rather than being a participant anchor in an association, the physical token acts as something that triggers or scopes applicability of other resources.

Traversal of Links

Where the link exists wholly in the digital realm, the idea of traversal is often quite transparent to the user. Taking the Web as an example, a user would be reading an electronic document that has a highlighted link anchor in it. They click on the link anchor initiating the traversal and are transported to the destination as it appears on the screen in front of them, usually replacing the previous document, presenting the 'illusion' of travel. This is automatic traversal in its purest sense. The user needs to do no more than click on the link anchor and perhaps blink and they are there.

In the physical realm, the onus is on the user more often than not to perform the traversal. For example, if we consider the index at the back of the book as a form of linkbase, the words are the sources of links. Next to the source anchor, for our benefit, has been written the destination anchor. It is down to us to resolve that anchor and then, if we choose to traverse the link, we have to turn to page 257, physically flipping the pages until we are presented with p257, the page referenced by the destination anchor printed in the index/linkbase at the back of the book.

3.3.2.2 *Link and Anchor Delivery*

In the case of the Web, anchors that are the source of links are passed in situ, along with the resources in which they are embedded, as are the links themselves. Both source and destination anchors of the Web's binary link model can be passed as references, although it is rare for authors of Web resources to name source anchors within documents, meaning that typically only whole-resource source anchors can be communicated. The DLS approach to anchor specification avoids this limitation by offering additional attributes when specifying anchors, for example, byte offsets into resource and 'selection text'.

In open hypermedia systems, both anchors and links can be passed around trivially. This is because the anchors are distinct and explicit references to objects and as such wholly separate from them. This is sometimes referred to as links and anchors being *first class* data, c.f. the Web's linking model being dependent on in situ data. As a consequence, links in open hypermedia

systems can also be communicated free of dependence on the resources that they associate.

As discussed above, physical anchors can be delivered by virtue of their digital representation. The scope of the representation, though, is an issue that is scenario dependent. For example, the validity of an anchor (or a link associating that anchor) being communicated to somewhere beyond the 'reach' of a resolution process is questionable.

Some applications may wish to be aware of the existence of the anchor or link and therefore infer some knowledge about the nature of the information space. For example, if an anchor to a physical document is present in associations (i.e. links) between other resources in a meeting (both physical and digital), and then the physical document is taken away from the meeting room, it is not clear whether the anchor, and therefore the links, should also be removed.

One option is that the associations become un-resolvable because the anchor cannot be resolved and therefore the links are never communicated by a link service. Another option is that the links are delivered irrespective of whether they are resolvable and it is up to the receiving applications to determine their validity. The latter approach pushes any awareness of the ability for link data to be resolved out to other components, simplifying the role of the link service; it also mimics existing DLS implementations that make no guarantees about the navigability of the links they communicate.

In the example given above, it may be the case that an application is aware-enough of the nature of the physical anchor's digital representation that it can be resolved into a different digital representation of the physical object that the anchor was originally bound to (e.g. an on-line version of a physical document). In this sense, the physical token is not so much an anchor as it is a digi-physi-link of its own right.

3.3.2.3 *Observation: Virtually Physical*

The realisation that arises from this analysis of Physical-Digital cross-boundary linking and anchoring is that it is largely irrelevant if a link associates anchors that are physical or digital in nature. That is, given an appropriate resolution mechanism, physical anchors can participate in hyperstructures. Likewise, physical links can be modelled digitally and therefore be participant in hypermedia systems, entangled in the hyperstructure.

With an appropriately open and shared link model, any link service that can deliver links and linkbases for digital resources can also be used for the exchange of physical and digital-physical links.

Placing this in the context of the scenarios that are defined in the next chapter, the DLS approach provides us the ability to incorporate resources such as physical documents, locations and furniture into our shared information space. This enables an extra dimension of links to be authored and exchanged.

The links could be authored manually, e.g. by providing a mechanism for tagging a physical document providing a digital anchor, or automatically, e.g. the digital observation of the collection of tagged documents on a table creates an association linking those documents together. As far as the underlying link service is concerned, however, these links appear just like any other. The characteristics pertaining their generation are irrelevant, however their location and delivery are not.

3.4 Summary

This chapter introduces a subset of the field of pervasive and ubiquitous computing that includes information systems, and how hypermedia techniques might be employed to provide discovery and navigation of information therein.

In discussing the notion of Physical-Digital linking, we have observed that the 'seamfulness' of information systems that incorporate notions and

representations of physical artefacts can largely be ignored, given appropriate consideration to the digital resolution of physical artefacts. This means that physical things in the environment can be incorporated into the information space and treated just as any other information resource, digital or otherwise.

The following chapter describes a series of scenarios that define an application space in pervasive and ubiquitous computing in which information discovery and navigation is desired. Later chapters document the development of the DLS approach to open hypermedia in order to cater for the issues and requirements arising from those scenarios.

Chapter 4 Scenarios

In order to scope the research activity, this chapter identifies three application scenarios in which a pervasive hypermedia-based information approach might provide benefit.

The methodology behind the selection of the scenarios is such that each should demonstrate different characterisations of activity, scale and content.

Activity. Task-less through collaborative tasks, and on to passive participation
Scale. Single-person or at most small groups, through collaborative groups, and on to larger scales
Content. Published materials such as posters and conference proceedings, through live to documents, also with various notions of physical presence as link anchors.

The scenarios below were chosen as a representative sample from the spectrum of workplace information systems in which information has tended to be pervasive in the sense of being all around, but not necessarily navigable or inter-linked within a hyperstructure.

The approach for each of these scenarios is to detail the various information resources present in the scenario arena, detail any management and ownership entities, and itemise likely interactions of users engaged in the scenario as it would be without the deployment of any pervasive information infrastructure. Where each scenario overlaps or builds on ones before it, repeated resources, issues and interactions are omitted.

Then, the scenarios are re-evaluated through 'hypermedia tinted glasses', with links, linkbases, link services and other information services identified. Resulting from this activity, the infrastructure requirements that would need to be met for services to be deployed are discussed.

4.1 The Corridor

The early motivation for this research stemmed from a thought experiment that arose from discussions on the different kinds of information that are all around us in everyday life, often serendipitously navigated, yet rarely explicitly 'linked'.

The notion was captured as the scenario where a person is walking along a corridor, and happens to glance at a poster on the wall. The text and images on the poster prompt the recollection of a number of associated things, people, places, events, etc. This could be thought of this as having a few themed collections of hypermedia links active in the mind.

In our scenario, a wearable computer behaves in a similar way, reifying and augmenting mental linkbases with various personal linkbases on the device or in the environment, for example, a linkbase associated with the user's personal hobbies and interests, one associated with their current tasks and one associated with the building they are in.

A mock-up experiment helped identify and explore the research issues that arise from the example, providing a vehicle to test the benefits and hindrances that 'extra' information due to the provision of a hypermedia-like enrichment services.

Figure 4-1: The Corridor Scenario

The experiment was performed in a typical corridor in a research lab, in our case, the corridor linking the entrance foyer of a University school building to the lifts to teaching and research floors – a communal and somewhat under-used space.

4.1.1 Resources

Within the space, there were a number of identified static information resources, dynamic resources, and 'utilities' providing conduits for additional information.

Static information resources

Signs. Rooms may have signs labelling the purpose of the physical space inside; Directional 'Emergency Exit' and 'Toilet' signs indicate the location of artefacts with function that are not immediately located

Posters. Posters describe events such as conferences, meetings, social activities, on which there are typically event descriptions, contact details for further information and registration, and (surprisingly frequently) Web URLs

Notices. Effectively scaled-down notices, there were also a number of small notices, such as reminder notes referring to events or activities not otherwise represented on the notice board, and 'Stickies' attached to other notices and posters serving to annotate

Dynamic information resources

Rolling display. A non-interactive display that scrolls notification and information about various local events, with an associated version on the Web for browsing back at one's desk

Intranet Web. The space is provisioned with wireless networking, enabling live Web browsing with one's hand-held or portable computer

People. There are three classes of 'live biological' information resources in the space:

Us. As users, with our own 'mental linkbases' active in our mind, reacting to information we see on the notices and signs, associating with other memories (relevant or otherwise)

Residents. People such as secretaries, or people with offices immediately adjoining the space

Passers-by. People in a similar circumstance to our user, using the corridor as a means to get somewhere else

4.1.2 Management and ownership

Once published by virtue of being pinned to a notice board or fixed to a wall, posters and notices become 'owned' by the building. Permanent fixtures such as signs and space labels, likewise.

Some of the information is slow moving in that it is conveyed by a permanent fixture such as a sign, whereas other information is ephemeral – once the notice is removed, the information and any other associations available due to it are no longer available.

Information held by people is 'managed' in the sense that they can choose how and when to articulate it, in response to any (or no) inquiry.

4.1.3 Example Interactions

Posters and notices are typically read-only media, and as such the principal interaction of users with information in this space is to retrieve facts (e.g. to a notepad, or just to memory), or references to facts (e.g. the contact details, the URL printed, the tear-off contact details).

As regards features such as signs, the interaction is physical navigation as a result of reference, should the thing being described by the sign be a desirable destination, or perhaps a casual mental note that the feature exists and its location remembered for later recall.

It is becoming increasingly common for users with portable networked devices (e.g. Palm Tungstens, laptops, or HP iPaqs) to browse serendipitously on the move. Where the space features wireless networking and the various notices and signs advertise additional information on the Web, this information can be browsed in situ, e.g. whilst waiting for the lift; or the reference captured for later perusal.

4.2 The Meeting Room

In this scenario, people convene in a meeting room with their computing devices and these devices form a shared workspace of resources for the meeting.

The photograph below is of a mock-up experiment that served to identify the themes of resources and nature of interactions as part of this scenario.

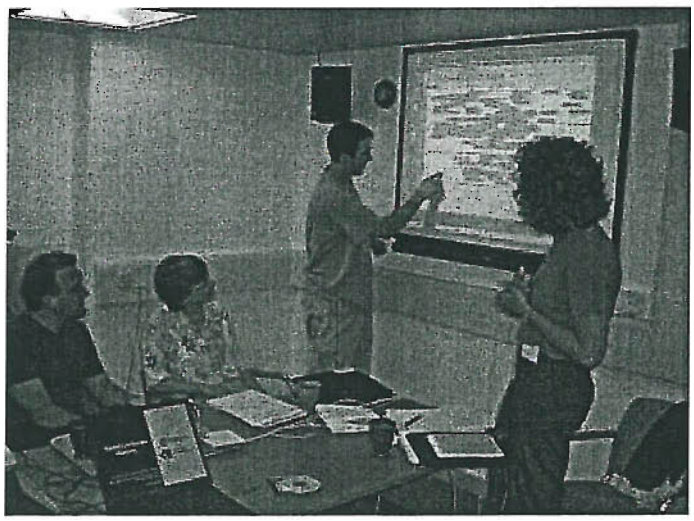

Figure 4-2: The Meeting Room Scenario

This scenario differs from the Corridor above in that it concerns less about

provisioned information and physicality, and more about resources that happen to be brought together within the context of a meeting. Where the Corridor might be considered as pervasive information in an 'embedded' context, this scenario regards pervasive as ubiquitous: a familiar use of hypermedia in a new context.

4.2.1 Resources

The room itself may offer infrastructure components, such as an addressable data projector, shared electronic whiteboard and audio devices. In terms of information resources, the following items have been drawn out in addition to those identified in the Corridor scenario above:

Static information resources

Papers and reports. This includes both the printed-out copies of resources brought by participants to the meeting, but also those copies that exist on laptops and other devices brought by the users, or accessible over the Internet

Posters and demos. Where the posters in the previous scenario typically relate to events, the kinds of posters observed in meeting rooms are of projects or products. In the mock-up experiment defining this scenario, the meeting room also included demonstrations of work that may also contribute to the information space, for example, as a related output from previous work relevant to the meeting

'Linkbases'. Participants may bring collections of links to other resources with them to the meeting. These collections may be manifested as 'bookmark' files (lists of Web-style Title/URL pairs); as BibTeX or Endnote citation dictionaries, e.g. referencing other papers or reports of pertinence to the meeting; or *bona fide* Open Hypermedia linkbases

Dynamic information resources

Collaboration tools. The meeting room might include technologies such as Smart Whiteboards that digitally capture their contents, resulting in additional resources for the information space. Also, the participants might use collaborative editing tools to author or revise documents during the meeting

'Live' documents. In addition to the collaborative editing discussed above, there are other resources that change throughout the duration of the meeting that might wish to be incorporated into the available information space. Examples include:

Annotations. Whether scribbled onto existing resources, or onto Stickies that are then attached to other documents, annotations are a primary information resource of interest

Minutes/Notes. Additionally, the rapporteur generating the notes that eventually become the minutes might capture annotations of later interest; likewise, a Compendium style live issue-modelling tool generates new information as a process of the meeting (Selvin, 2001)

4.2.2 Management and Ownership

Discrete resources can clearly and readily have their ownership identified – the participant that sourced the item can be considered the owner for the purposes of the scenario. However, resources that are generated as a result of the meeting, or developed during it, have a more complicated ownership issue.

It is clear that the intellectual ownership role would be well defined by the context of the meeting. For example, in the case of a meeting whose purpose is to define the structure of a paper or report, and perhaps delegate sections of it to different authors, the ownership role would lie with the authors. In the case of a meeting discussing a meeting between people within a project or

institution where the intended output from the meeting is not a particular document (e.g. a research development meeting), the ownership of any resources generated during that meeting is unclear, most likely defaulting to the person that initiated the resource's creation.

The management of information resources follows a similar pattern to that observed for 'mental' resources from the Corridor scenario in that each individual (or processes acting on their behalf) would be responsible for managing their own resources, choosing when to reveal new resources or manipulate others.

4.2.3 Example Interactions

Some resources in the space, for example the Agenda, might be referenced by different parties at different times without update, in a similar manner to posters and events of the Corridor scenario. The nature of the resources, though, suggest that it is more likely for them to be exchanged between participants, e.g. a copy of a report or the last meeting's minutes distributed amongst some members of the group.

Typical interactions in the previous scenario did not affect the resources in the scenario. New annotations or resources might be created, other associations discovered, but no substantial change to the resources already present. In the case of the Meeting Room, however, it may be the case that documents brought into the space are edited as a process of the event (e.g. a report writing session, or a review critique).

Annotations in this scenario manifest themselves in a number of different ways. A public annotation of a report might persist beyond the meeting (e.g. a reflection on a particular section of a report provided for the meeting); or it may be a transient marker for later discussion, augmenting the agenda resource and therefore not necessary beyond the end of the meeting. Likewise for private annotations, for example, a 'note to self' regarding an action that needs to be completed relating to a particular item brought up in the meeting.

In some instances, the annotation might a new resource in its own right (e.g. the 'note to self'), associated to an existing resource. In other cases, the annotation might be a characteristic of an association between two resources, for example, a note attached to a paper that suggests somewhere else to look for related material.

Where there are more participants in the scenario, there is more scope for exchange of information, whether it is whole-resource exchanges (e.g. giving a report to someone), partial exchanges, or mere reference exchange – akin to swapping bookmarks.

4.3 A Conference

A larger-scale scenario than above is that of a Conference room. This scenario shares many characteristics of the Meeting Room above, but with many more participants.

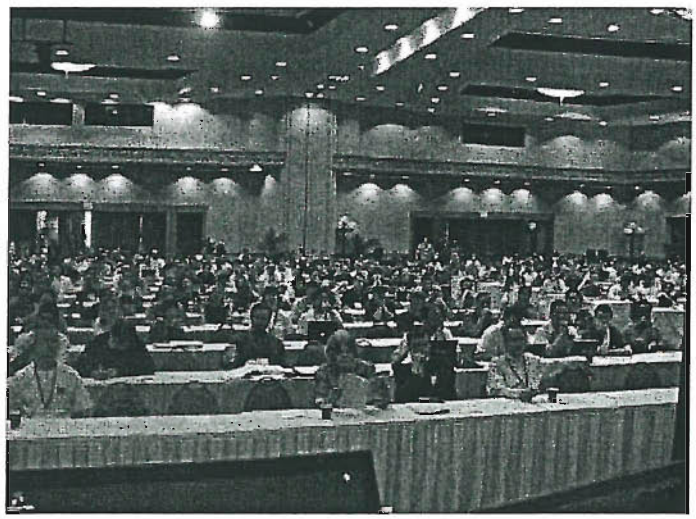

Figure 4-3: The Conference Scenario

4.3.1 Resources

This scenario shares a very similar set of resources to the Meeting Room scenario, only on a greater scale. The nature of the resources is typically read-only, e.g. proceedings, posters, demonstrations, and the only new resources generated being either annotations or associations.

Static information resources

Proceedings. Read-only corpus of published papers and posters forming the tangible output from the event. In the Meeting Room scenario, there is little likelihood that multiple instances of the same resource (i.e. paper) exist in the scenario, however with the case of the Conference, almost every participant will appear with a common set of resources

Notices. Similar to the posters and notices of the Corridor scenario, except that much of the notice content refers to more tightly related theme (e.g. local administrivia, future workshops or conferences within the same community)

Conference Web site. It has become the norm for a conference to have an associated Web presence, not only for marketing purposes, but also as a coordinating resource for logistics (e.g. travel directions, accommodation) and additional information that, like the 'notices' resources, may have local relevance for people at the event

Dynamic information resources

Associations. Whilst also a potential resource in the Meeting Room scenario, associations between information delivered as part of the Conference, and other information either also within the scenario, or brought in by other participants feature as a dynamic resource

Annotations. The nature of the Conference setting is that participants are exchanging ideas and discussing work, and therefore annotations of the proceedings are likely to be prevalent

4.3.2 Management and ownership

This scenario introduces no new issues of management and ownership.

4.3.3 Example Interactions

For the purposes of this scenario, interaction of information in conferences is split into two modalities. First is the plenary session, where participants annotate the event, whether in-situ on their copy of the proceedings, or as new resources in their own right associated with the paper being presented. The second modality is a more social variation of the Corridor scenario, where participants walk around and meet other delegates and interact with posters being presented as part of the event.

4.4 Through Hypermedia-tinted Glasses

This section considers how a pervasive hypermedia-oriented information systems middleware might benefit each of the scenarios.

4.4.1 Corridor

There are various ways in which the different information resources in this scenario could have digital counterparts that could participate in a hypermedia system, some of which have been discussed in the previous chapter.

By way of explicit example, posters and signs could have infrared beacons that emit vCards (IMC, 1996) that resolve to URLs at which additional information could be discovered, as in the Hypertag system[2].

Alternatively, physical anchors such as bar codes and RFID tags could be deployed that, when scanned or received by an appropriate receiving device, can be resolved into digital anchors, whether through the use of a third party resolution service as in Kindberg's Pulp Computing (Kindberg, 2003), or directly using the token's identifier as an anchor itself.

Techniques such as Pin&Play (Laerhoven, 2003) provide mechanisms to tag things such as notices and posters with digital IDs, in addition to a visual

[2] Hypertag: http://www.hypertag.org/

interface (flashing LED) that can be stimulated as a result of resource location query, for example, as a target anchor for a link.

If one were to assume that the various information resources had a digital counterpart, then, the following hypermedia artefacts can be identified:

Space associations. Door signs associate a physical space (the space behind the door) with the concept of the purpose of the room, whether that be a person or group of people in the case of an office, or a role (e.g. 'Reception'); Direction signs (such as 'Toilets →') are similar links, although their destination anchor is requires a more involved resolution process, for there may not be an explicit space being referenced. The act of traversing the 'link' does not result in the navigator arriving at the anchor, but rather at a place from which further navigation should result in arrival.

Genuine associative links. Some of the artefacts within the space already feature genuine associative links (De Rose, 1989) in that they detail locations at which additional resources containing information pertaining to the notice or poster's content can be found. A software process representing the physical poster and its information could also maintain a linkbase that includes these bona fide associations between concepts. These associations would require authoring, for example, by the person sticking the poster or notice on the wall.

An example of associative links that could be either created automatically (e.g. by image analysis extracting text and recognising the relevant sections) or explicitly authored are links that associate methods of contacting someone regarding the resource. Various URI schemes have been standardised for the mark-up of contact information. Email contact points can be referenced with `mailto:` URLs (Hoffman, 1998), and telephone and fax with `tel:` and `fax:` respectively (Vaha-Sipila, 2000).

Annotations. Sticky notes attached to notices and signs that annotate their content also add to the hyperstructure of the scenario, although their content is somewhat difficult to encapsulate digitally. The fact that a physical annotation exists could be sensed if tagged using one of the techniques

already discussed, and therefore the physical anchor of the annotation included in the hyperstructure or scribed on augmented and sensable paper (Frohlich, 2002), or on an interactive whiteboard such as the one in Figure 4-2 on page 77.

An additional mechanism would be required for creating the association between the tagged-note that represents the annotation and the tagged-poster or notice being annotated. Virtual annotations, as discussed in (schraefel, 2004), would provide navigable and digitally retrievable content associated with the poster being annotated.

In addition to the types of link above that are derived from or explicitly added to the physical environment, there are also the various themed, task-specific, social, and miscellaneous linkbases that are associated with the users, whether on their mobile devices, or present on the building's computer infrastructure. It is a natural requirement that their incorporation into the available information space is enabled by any adaptive infrastructure developed in addition to support for the interactions observed above.

4.4.2 Meeting Room

An example of a complex example interaction in the meeting room scenario, augmented by an enabling hypermedia system infrastructure would be:

> Person A makes available their resources – document, slides, video clips and themed collections of links (linkbase fragments) – on the topic of Contextual Hypermedia. Person B, having an interest in the development of contextual applications using Java, deploys a service that continually surveys the information space for related resources. A third person, Person C, arrives in the meeting space with a linkbase that has associations regarding a hypermedia workshop's proceedings, which includes a paper by Danius Michaelides *et al.* on the Auld Leaky contextual link service. Person D, having performed a literature review as part of their PhD programme, has an annotated link from an 'unknown' paper to a Java implementation of Auld Leaky.

A desirable outcome from this particular example would be for Person B to discover the Java implementation of Auld Leaky, Person C to discover a video demo further describing the notion of Contextual Hypermedia in Person A's resource-base, and for Person D to 'fix' their badly associated unknown paper reference.

In addition, any pre-existing hyperstructure across those resources should be available to all of the participants, with an appropriate level of user control, such that they can be resourced by link services and applied to the resources available. This may include, for example, local linkbases that tie together annotations of meeting notes in a local minute-pool, or generic links that might be applicable when a resource is identified as being themed on a user's interest.

It is likely that any pre-existing linkbases brought to the meeting place will refer to resources 'out there' on the Web, whereas instances of those resources referenced may actually exist locally, and so a useful function of an enabling infrastructure to capture that notion, and make available the local instance when appropriate.

Even if participants of the meeting do not use the resources explicitly, automated processes may wish to – perhaps, trivially, for search purposes, or in general to answer a context-sensitive query, such as automatically harvesting links associated with a particular theme.

Once the meeting is over, the hyperstructure that has been created in the meeting, through the amalgamation of linkbases, documents and interactions (the ad hoc information space) is deconstructed, any newly created structures or documents captured somehow for later dissemination.

Likewise, should the meeting break out into smaller groups, new hyperstructures could be created, perhaps re-using fragments of the original structure, and a filtration process required that reflects the nature of the (now partitioned) information space.

The fact that new resources can be created on the fly introduces an issue regarding persistence. One approach for making newly created resources persistent would be for copies to be published to a globally accessible store, for example, a DAV-enabled Web server (Whitehead, 1999), and then anchors that were created during the scenario that reference the local resource updated to reflect the revised, 'permanent' location of the resource.

A similar approach could be taken for resources that are modified during the meeting, however, consideration then for version control would be required, especially in instances where the semantic meaning of the resource was edited sufficiently such that its relationship with other resources was changed.

4.4.3 Conference

In the Conference scenario, each delegate could have digital copies of the proceedings and associated materials on their laptops or on a local organiser-provisioned Web server. Each delegate's personal view of the Conference information space would then be built up by the combination of their collection of relevant bookmarks and personal linkbases, applied to the other resources present in the scenario.

Extending the model of the meeting room above, the linkbases of other participants would also provide potential hyperstructure sources. The outcome here, with more participants than with the Meeting Room scenario, could potentially be hundreds of different views of the same set of core resources, with many additional (and navigable) sources that would otherwise be unavailable. By enabling these views to be shared, integrated and captured, new interpretations and additional value could be gleaned from the conference-provisioned and hyperstructurally sparse resources.

Notes and annotations – in particular, links – made in real-time during presentations enable the hyperstructure to evolve in a collaborative fashion. An example of the use of a tool to create hyperstructure as a side effect of natural (but 'online') communication is given in section 5.2.6.5 below.

4.4.4 Physical resource participation

There is a risk that significant effort may be put in to creating a wholly encapsulating and precise digital representation of the physical environment involved in the scenarios, and that the return for that effort might not be satisfactory. Whilst this may be a laudable goal in some applications, it is not necessarily the suggestion of this thesis.

Rather, we observe that physical, situated resources that otherwise have no 'digital' or navigable presence *could* be included in the information space, and therefore an agenda for the research is how their inclusion might be achieved.

There are a number of cases of some level of digital presence for physical information resources:

- Digital representations of physical artefacts in the scenario already exist, but no standard mechanism for interacting with them or incorporating them is available, or they are in an inappropriate form; e.g. on-line telephone directories mapping people to places.

- Digital representations could be automatically generated by web-cam image analysis or other detection technology; e.g. a content-based technique such as developed with the Microcosm Architecture for Video, Image and Sound (MAVIS) architecture (Lewis, 1996)

- Digital representations that require explicit and manual authoring; for example a tagging mechanism for notices pinned to a notice board, and the generation of links anchored on that physical tag

Common to all of these approaches to participation, there is the notion of a physical anchor that can be captured and rendered digitally. That anchor can then participate in link associations, which then only make sense (are resolvable) within a context that understands the nature of the anchors.

For instance, the physical token attached to a First Aider sign acting as a physical anchor might be the Code-39 barcode '*S59471101*', as shown below. When scanned by a user, the token is resolved into a digital anchor, which can then be used as part of a query for associations regarding that particular sign, whether directly (links anchored on the digital representation's concept) or indirectly (digital representation is resolved to a concept, which is then used as a query component).

Figure 4-4: Physical Token used as an Anchor

An observation here is that this required an explicit action to discover the presence of the anchor. Similarly with a passive sensing technique such as RFID, local presence triggers anchor discovery by the user.

Other modalities would include users explicitly querying the environment for anchors representing resources (physical or otherwise) in the space, for example, by interacting with a hand-held or infrastructure provisioned 'sign'. Alternatively, anchors could be discovered as a side effect of other queries on other known anchors in the local space. For example, having discovered a poster and queried for associations regarding that poster, an anchor representing some other local physical artefact could be discovered as part of the result set.

These interactions are familiar to DLS-based open hypermedia systems, in that typical query interactions with the DLS are:

- Where a user already has a resource and explicitly queries for links pertaining to the resource as a whole, or individual anchors within, from which may result new anchors on the same resource or new anchors on new resources (CGI-mode, explicit DLS interaction)

- Where a user retrieves a new resource and as a result of that retrieval process, the representation of the resource is enriched with additional

anchors and associations from those anchors (web-proxy applying hyperstructure on-the-fly, implicit DLS interaction)

This observation reinforces the selection of a DLS-like infrastructure for discovering and navigating information in hybrid environments.

4.5 Common Requirements

The notion emerging from these scenarios is that the hypermedia middleware, fundamentally the link service, needs to support ad hoc hyperstructure formation by supporting a degree of impromptu networking and the combination, sharing and open access of linkbases, spontaneously available, in order to navigate information that is incidentally available due to the people and devices present in the environment.

When considering how the scenarios might be realised, a number of common requirements are observed at two levels: the network infrastructure permitting devices to intercommunicate, and the adaptive software infrastructure enabling hypermedia and resource sharing services.

The sections that follow detail issues and set out the agenda for the experiments in later chapters that investigate appropriate architectures.

4.5.1 Networking

Beneath any software infrastructure, a notable consequence of the evolution towards devices and computation platforms everywhere is that there will be an associated growth in the diversity of networking approaches between devices.

Wired networks around the office are already at Gigabit-per-second speeds using inexpensive hardware, and Internet connectivity has become ubiquitous in the home with deployment of always-connected technologies such as Cable Modems and Asynchronous Digital Subscriber Line (ADSL) networking (for Personal Computers, at least). Wired networks prohibit portability, however the affordability of wireless access networks has become sufficiently

accessible that their presence is becoming commonplace in office buildings, campuses, and the high street.

Wireless networking technologies are emerging at speeds that were deemed more than acceptable for wired networks ten years ago at rates of two to ten megabits per second, and more recent technologies achieving 54Mbps (IEEE802, 2003). This is in part due to a more open approach by the communications industry through their cooperation in standards activities such as the Internet Engineering Task Force (IETF) and by forming powerful consortia.

Three examples of the more important consortia in this space are the 3rd Generation Partnership Project[3] for third generation cellular networks using Wideband-Code Division Multiple Access (W-CDMA); The Bluetooth Special Interest Group[4] for low-cost, low-power technology labelled as "personal area networking"; and HomeRF[5], who are standardising a protocol called Shared Wireless Access Protocol (SWAP) which merges DEC-T digital telephony with two-megabit data communications.

Given all these access technologies, and the need for more and more heterogeneous devices to intercommunicate, it stands to reason that they should all in some form be able to speak a common 'language'. Research by Karim and Hovell has observed that the Internet Protocol (IP) has emerged as the most commonly understood protocol for internetworking (Karim, 1999).

IP is the protocol by which the public Internet is accessed, and it is the network protocol being designed-in as standard in emerging mobile telephony standards replacing GSM (Kempf, 2000), and in embedded systems replacing technologies such as Component-Area networking (CAN).

---

[3] 3rd Generation Partnership Programme: http://www.3gpp.org/

[4] Bluetooth Special Interest Group: http://www.bluetooth.org/

[5] HomeRF Working Group: http://www.homerf.org/

In brainstorming the target scenarios, we consider that the features a network layer (i.e. IP) must offer to devices such that they can operate when truly ubiquitous include:

- Addressability. *There must be enough unique, globally routable addresses to assign to all devices likely to need to communicate, especially considering that devices may be always-on, and therefore need to be always reachable*
- (near) Zero-configuration. *Devices should possess the ability to seamlessly join and leave the network, leaving participant applications intact; without manual configuration or intervention where possible*
- Multi-modal. *Devices must be able to operate across different network media, e.g. a user unplugging a laptop from a fixed Ethernet network switching to 802.11 wireless networking. Devices should be able to operate and select the most appropriate network (whether due to application requirements or topological suitability) when more than one medium is available and applicable at any given time*
- Routing. *The routing protocols must scale to incorporate uniquely addressable routing. That is, no private networks or Network-Address Translation (NATs) that impede end-to-end connectivity between applications*
- Mobility. *Devices should be able to roam freely, connecting at topologically or physically different locations, thus the underlying network should support mobility in a scalable, robust fashion. Ideally, the device should remain uniquely addressable by a consistent identifier (IP address or labelled name) whilst mobile*
- Security. *Whether due to being in a sensitive environment, or to enable accounting, secure (authenticated, authorised and integrity-assured) communications should be permitted where required*
- Quality of Service. *Methods should be available to ensure some notion of assured quality of service in data delivery, for example, to enable end-to-end timely delivery of media streams, such as voice (e.g. Voice-over-IP, VoIP)*

(Kolon, 1999)

Some of these identified challenges can be ignored for the purposes of the more locally focused scenarios. However, we feel it important to keep the issues of a more global scale scenario in mind when identifying candidate

solutions, as it is inevitably the case that utility gleaned from local-area applications will be applied to larger-scale systems at a later date.

In considering these network requirements, the following observations regarding emerging technologies were noted:

4.5.1.1 Addressing and Routing

The current commonly used version of the Internet Protocol, IPv4, is not scalable to the level needed to satisfy the demands placed upon it by the predicted explosion in number of Internet-connected and potentially globally-reachable devices. IPv4 has a 32-bit address space, and therefore has a theoretical maximum number of addressable entities in the order of four billion. However, the topology of the Internet is devised around allocated clusters of addresses, 'subnets', with routing protocols developed such that packets can be forwarded between different networks. The allocation schemes used has seen this address space partitioned up, with some addresses effectively 'lost' for use by devices.

Solutions such as Classless Inter-Domain Routing (CIDR) (Fuller, 1993) and Network Address Translation (NAT) (Egevang, 1994) have served to defer the point at which no new devices can participate in a global Internet, however they do come with associated costs. In the case of NATs, the end-to-end nature of connections between devices is lost in that one or both participants of a connection cannot be certain that the address reported in a connection (e.g. IP source address header) will be the same device from connection to connection.

This is especially problematic for protocols that rely on a definable end-to-end path, such as voice conferencing with the H.323 protocol. There is also address space wastage due to designation of different blocks of addresses as for private (i.e. non-routable) use (Rekhter, 1996). Where private addresses such as those offered by NAT defined in RFC1918 could be used within the context of the local scenarios identified here, for example one cloud of devices within a single building, if communications are required to remote networks then globally routable addresses are required.

4.5.2 IPv6

The next generation Internet Protocol, IPv6 (Loshin, 1999) features solutions for many of the issues noted above, most prominently its 128-bit addresses and more efficient recommended address allocation policies offer space for huge numbers of globally reachable devices. The core specifications of IPv6 have been completed since 1999, and there is ongoing activity within the IETF (Internet Engineering Task Force) to further refine the novel features offered by the protocol.

An impromptu model of interconnection is supported in IPv6 through the specification of stateless auto-configuration and automated neighbour discovery protocols. A device connecting to an IPv6 network is able to configure its own network address on the fly, and can detect its local network gateway via router advertisements, without user intervention.

The protocol also supports the notion of a 'link-local' network, where nodes are guaranteed to be able to be able to address up to 2^{112} local hosts uniquely, and is ideally suited for use in the case of isolated networks where there is no gateway router to an internet. For example, this enables two mobile devices that happen to have migrated to the same local network to establish a link local connection (i.e. communicate directly on the same link without having to propagate packets back to their respective home networks), even if their home networks are physically far apart.

Security is mandated in IPv6 to the extent that all devices must support a profile of IPsec, the mechanism for securing the network layer through encryption and/or authenticated encapsulation. The choice of security model at the network layer has a direct impact on the application layer. For example, with a known connection semantics - for example, X.509 certificate and session encryption - authentication at the application layer may not be needed.

4.5.3 Mobility

Mobility can be considered a characteristic at three levels: device, service and person. Device mobility is the change in location of a device whilst maintaining active communication (same device, different point on network); Service mobility captures the idea of a service moving between devices, connected in different places of the network; and Personal mobility is the notion of a person moving and appearing at a different device, registering for a remote service on a device at the original location (Yang, 1999; Schultzrinne 1996).

The notion of nomadicity, as defined by Bagrodia (Bagrodia, 1995), diffuses the levels of mobility into the need to provide transparent, integrated and convenient computing and communication services to roaming users.

Where 'Wireless' and 'Mobility' share many synergistic features, they are not the same thing. It is possible to have wireless devices that do not move, just as it is possible to move wired devices between different patch-points on a wall. The difference comes when the network topology changes as a result of the device moving.

Many solutions to device mobility are to provide Virtual Networks overlaid onto real physical networks (Ioannidis, 1991). Overlay networks abstract the underlying topology, and are often used to develop or trial new services that are not available in the current networking infrastructure, as well as providing improved performance for the existing services (Amir, 2002). However, this technique typically requires additional infrastructure at the gateways between the various underlying networks, and therefore incurs an additional administrative burden.

Using the overlay technique, devices connecting with different media types can appear 'local' to each other within a local environment, even if their underlying network connectivity is vastly different. Consider a laptop connected over GSM to an ISP in the same room as a hand-held connected via a Bluetooth access point to the local LAN. For the purposes of the local scenario, the two devices would ideally be situated on the same local

network. An overlay network can be deployed such that this is the case, but at considerable expense: every packet of traffic destined for the laptop would need to be encapsulated, forwarded to the (remote) ISP and then on to the laptop over the GSM network, introducing considerable cost in latency, throughput, and cost.

Additionally, technologies such as proxy-ARP, where nodes act as bridges between different network segments, can create the illusion of a flat IP network that includes devices that would otherwise appear off-link and therefore not local.

4.5.4 Service Discovery

Acknowledging the fact that IP appears to be pervading all internetworking systems (Karim, 1999), and that every network-able device will have an IP address of some description, even with the issues of self-assignment, self-configuration, mobile, secure, etc. resolved, the technique by which services within the network are discovered is still an issue.

Service broker technology, such as Salutation (Salutation, 1998), Jini (Scheifler, 2000) or the IETF's Service Location Protocol (SLP) (Guttman, 1999), define service discovery semantics that may be applicable in the cases where their specification encompasses the available technology. However, there are certain assumptions - such as the presence of a Java virtual machine - that may render such approaches impossible. A sample of these technologies is discussed further in the prototyping section in the next chapter.

A common approach to service discovery is for clients to query a service directory with a template of attributes about a service that they wish to discover. Based on the description, an address of, handle for, or instance of a matching service or its proxy is returned, at which point clients can communicate with their intended service. There is an associated bootstrap issue here in that the client still has to discover the service directory agent.

Typical approaches to this problem include the notion of a multi-/broadcast 'ping', which are sufficiently lightweight as to be simple to implement and able to discovery the 'generic' service directory agent (e.g. as in SLPv2). The notion here being that the overhead of finding a service directory agent is beneficial with regards the specificity that the discovery protocol can provide when searching specific service instances. That is, it is simple to discover a simple service (a service directory agent), but more complicated to discover a specific instance of a particular service (a particular data projector with precise display characteristics in a known physical location).

José argues in (José, 1999) that SLP does not scale to wide area systems, nor does it convey adequate information for the physical location of a service when that data is required. Jini is a Java-based solution that serves as a broker for services and devices, but has the requirement that there is a Java virtual machine present on all devices that wish to talk with it.

Scheifler (Scheifler, 2000) claims that being unabashedly Java-centric will simplify all systems, however the ubiquity of Java execution environments is neither apparent in current technology nor does it appear likely to become so. Where Jini was initially touted for multi-platform development, all visible efforts appear to be strictly Java only.

The Salutation protocol aims to solve all service discovery problems in an interoperable way through being an open standard funded by major corporations, developed in tandem with the Bluetooth architecture for wireless communication between arbitrary devices. However, there are as yet no functioning applications using Salutation.

Where less comprehensive or distinguished service discovery is required (e.g. just any instance of a service, or one whose name but not address is known, regardless of particular attributes), less involved approaches can be taken, such as found in early systems with a discovery aspect, including AppleTalk NBP or NetBIOS's SMB (Williams, 2002).

These technologies used a technique similar to the directory agent discovery above, where requesting applications dispatch a datagram to multiple hosts either by broadcast ('all hosts on a subnet') or multicast ('all hosts belonging to a group'), and only those hosts recognising the correct structure of the packet and thus capable of providing the service required would respond.

The 'Anycast' addressing scheme is a mechanism that enables a number of different, topologically unrelated nodes to share the same IP address. A device can then send a packet to any one member of the group, where the topologically 'closest' node as determined by the underlying routing protocol would receive the communication (Partridge, 1993).

4.5.5 Automated discovery and configuration (Zeroconf)

One approach to both spontaneous networking and service (component) discovery is emerging from the local area network administration activities of the Internet standards body, the IETF. The ZeroConf (Williams, 2002) working group are defining a set of protocols that enable devices to configure network interfaces, perform basic name-to-address mapping and a rudimentary level of local service discovery.

The charter of the Zeroconf Working Group is to make it possible to take two or more computers, connect them with a crossover Ethernet cable or 'plug' them both on to the same wireless LAN, and have them communicate usefully using IP, without needing an administrator to intervene.

Non-IP-based solutions already exist for this type of on-demand networking, for example protocols such AppleTalk and Microsoft NetBIOS currently handle this very well. However, as observed earlier, IP networking has become the de facto standard. With the introduction of increasing numbers of networked devices, not necessarily featuring a user interface through which configuration could be performed, some standardised level of provision for this type of networking using IP has been identified by the community as a requirement.

One key difficulty with this type of scenario is that it requires application developers to support one protocol for wide-area communication, and a different protocol for local communication. Using an example from the working group's charter document, a game developer writing a multi-player game will usually support IP to allow game-play across the Internet. A developer selling a game for $50 does not have the technical support budget to provide telephone support for people trying to configure their own private IP network at home, so for the sake of ease-of-use, that developer also has to support AppleTalk (in the Macintosh version) and NETBIOS (in the Windows version).

Unfortunately, even after doing all that work they have not solved their problem, because if someone with a Mac laptop wants to play a network game with their friend who has a Windows laptop, they are still in the position of having to set up their own IP network, because IP is the only cross-platform protocol their two machines have in common. Network printer vendors have the same multi-protocol support issues (Cheshire, 2003). The community determined that it would be much better if a single common protocol worked in all environments, thus the Zeroconf working group.

To achieve this local-area small-network functionality in IP, the working group are focussed on four main areas of work:

- Mechanisms to allocate addresses without the presence of a DHCP server (Dynamic Host Configuration Protocol – an administered service that can offer address, routing and other application configuration information to nodes of a network, typically scoped to enterprise networks)

- Mechanisms to resolve names to local-area IP addresses and back again without pre-configured DNS server (Domain Name Service – an administered service that, Internet-wide, provides the translation on behalf of applications)

- Mechanisms to discover services, such as printers, without the use of a directory server

- Mechanisms to enable the participation in multicast networks without a MADCAP server (Multicast Address Dynamic Client Allocation Protocol, a protocol similar to DHCP for address allocation, but concerning zoned-multicast addresses for applications).

A final requirement is that the solutions in the four areas must coexist gracefully with larger configured networks, that the developed Zeroconf protocols cause no harm to the network when a Zeroconf machine is plugged into a large network.

4.5.5.1 mDNS

Recent standards activity on zero configuration networks has settled on a local-area solution for non-infrastructure provisioned DNS-style resource record query in a technique called multicast-DNS, mDNS (Cheshire, 2003b). Development of mDNS was motivated by the observation that human-readable labels are more usable than remembering IP addresses, especially in the case of impromptu networks where these addresses are likely to be short-lived.

Each node in a zeroconf environment can run an mDNS service that is authoritative for its own name label only, as a member of a new, dedicated and geography-neutral top-level domain in the DNS naming hierarchy, *local*. mDNS servers join a well-known multicast group that is administratively scoped to the local network, and strictly service queries for their dedicated part of the naming hierarchy only.

In the example below, an application on the node with the (user-assigned, or randomly generated) label 'Weed' wishes to navigate a web page on the node 'Bill'. Weed's mDNS resolver queries for an address record for the label `bill.local` (1). Both the Bill and Ben nodes receive the query, for their mDNS processes are in the same multicast group, but only Bill replies with a unicast

response to Weed (2), because Bill's mDNS instance is the only authoritative node for its own domain label.

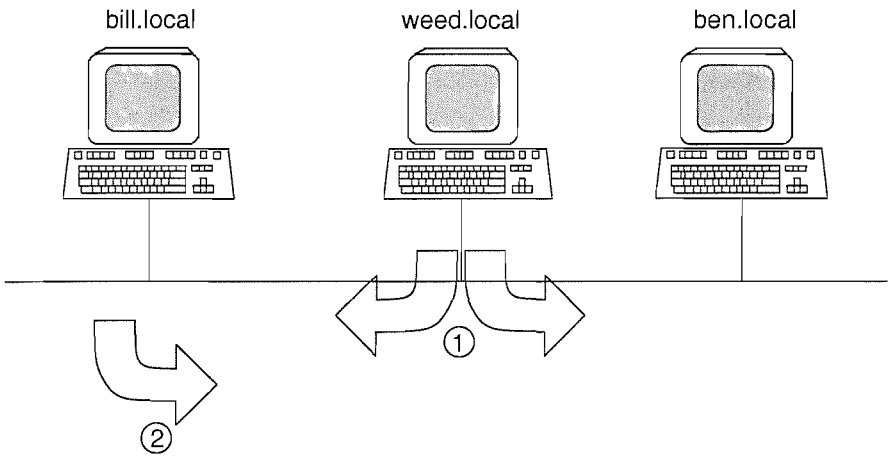

Figure 4-5: mDNS Example Interaction

In parallel to the development of the zeroconf solutions for symbol-to-address mapping, the standards community has also been specifying mechanisms for service discovery using the Domain Name Service as a resource base in a protocol called DNS-SD (Cheshire, 2003a). DNS-SD complements mDNS and enables nodes to discover instances of services in link-local network, providing a lightweight, ubiquitous (where the modern operating systems support) mechanism to find local instances of local network services for local processes.

4.5.6 Summary

Network infrastructures for pervasive computing environments similar to those identified in the scenarios of interest are already the focus of on-going research, if indirectly in some cases.

The assumption carried forward into the research of the adaptive software infrastructures that enable the kinds of interactions we require in realising the scenarios is that the underlying network is a flat, IP-based network.

Node IP addresses are to be automatically configured whether by zeroconf link-local means or by DHCP servers provisioned as infrastructure by the

hosting site. The mDNS responder approach is assumed for link local node name-to-IP address resolution and local service discovery.

4.6 Software Infrastructure

The fact that participants, their data and services, can join and leave the environment at any time has dramatic implications on the nature of the hyperstructures that might be available for navigation in the pervasive setting.

The set of applications and resources that are available at any particular time is unknown and unpredictable; the same query for a document or link resolution may yield different results when repeated moments later.

Hypermedia in the Pervasive setting is *probabilistic*, as opposed to the *deterministic* nature of traditional hypermedia systems – pure coincidence determines what resources are available to participants at any particular time.

By dynamically monitoring the set of resources available for navigation, and provisioning facilities to automatically generate new hyperlinks between them, it will be possible to create a localised and transient Web on the fly.

In this section we identify requirements of the software infrastructure resulting from analysis of the identified scenarios.

4.6.1 Requirements

An analysis of the scenarios from a hypermedia perspective suggests that there are six primary actions on resources in the information space:

1. Reference. *Read-only access to a resource or service*
2. Exchange. *Copy a resource from a peer to a local device*
3. Create. *Create a new resource*
4. Edit-copy. *Copy a resource and edit a new version*
5. Edit-replace. *Edit a resource in situ*

6. Publish. *Make available a resource to others*

Also, the observed trend of management and ownership issues can be summarised as 'to each their own, and perhaps yours too'. Resources brought to the scenario by an entity are owned and managed by that entity or a software process on their behalf. Ownership should be transferable, much as physically giving an item to someone else would be. Likewise, resources created as a process of the scenario should be owned by the group responsible for creating them.

These issues are captured as the following infrastructure requirements:

- Navigability of resources given their reference in links
- Naming and reference of local resources
- Consideration of different versions or representations of resources
- Discovery of resources in local context
- Secured and appropriate access to resources
- Facilitate extensibility through open-ness
- Publish linkbases so that others can query their content
- Allow fragmentation of linkbases to afford distribution
- Support linkbase fragment mobility as a route to optimised interactions
- Minimise configuration and administration tasks to maximise perceived utility
- Provide means of making persistent transient or generated resources

4.6.1.1 Provision Navigability

In the target scenarios, the resources that may be present on users' devices may include copies of resources that were originally published on the World Wide Web, on intranets or on personal file stores. 'Copies' here can be interpreted as verbatim, reformatted, summarised, annotated or augmented versions of resources.

The de facto method of choice for addressing these resources in their 'home' context is typically through URLs, or naming services that resolve well-known URLs such as PURL (Weibel, 1999). These naming approaches all refer

to the original context of the resources, not the copy that happens to be local and therefore more accessible, perhaps in a more appropriate format.

This suggests a requirement for considering navigability when resources may not be present locally. Should links referencing the current unavailable resource be returned as a valid query response? If so, how might applications render the fact that an association exists, but one or more of its participants are not navigable?

Local naming

An associated issue is that concerning local copies of globally available resources. Typically, the source and destination anchors of links are authored to refer to the documents in their 'global' context. However, in the scenarios alluded to here, those URLs may be meaningless as locators.

Providing a centralised resource naming service, like the one suggested by Tzagarakis (Tzagarakis, 2000), is not suitable given the ad hoc nature of the network of peers. Rather, a decentralised service that maps global URLs to identifiers to resources that are presently available may be of benefit, perhaps on a per-participant basis. For example, one participant does not care about the navigability of the anchor because it is simply using it as an opaque token. Another participant might, however, want to be able to retrieve the (local) resource in the association.

4.6.1.2 Provision Versioning and Representation

A resource on the Web is likely to be published in one primary format; in pervasive environments this is not always the case. Different device capabilities (screen size, colour depths, disk space, etc.) suggest that it is more likely that different representations of the same resource will be available in the local information environment. Another factor is that the local instance of the resource may have different content compared to the published version, whether due to being an earlier (or later) revision, having been augmented with user annotations, or perhaps totally re-written. These different instances

of the same information may have the same name and metadata as the publicly available, primary copy.

The distinction between the resources may need to be made explicit, else there is a risk of misrepresentation of information gleaned from the local copies; for example, a local version of a document may have been edited such that it is semantically different from the original version and therefore the fidelity of the information should be questioned. Mechanisms in pervasive information environments are required to ensure that users are made aware of these confidence-limiting differences.

4.6.1.3 Provision Resource Discovery

Aside from techniques of manually keeping indexes of locations of documents (e.g. bookmarks), a popular approach for resource discovery has been to build large-scale caches of static web content, and then provide querying services across that cached, indexed data, for example Google (google.com). This Search Engine approach has a varying degree of success when searching for specific resources, and is often constrained by chance that a querying user chooses the correct set of search criteria to discover the desired resource. This is a valid approach to building a 'web on the fly' but makes assumptions about the ability to search dynamically.

Activities such as the Semantic Web (Berners-Lee, 2001) are addressing the notion of assistive discovery using knowledge techniques, alluding to a "Give me what I meant, not what I asked for" approach to searching. At the same time, the Web is moving toward process-to-process communication in addition to process-to-human, through activities such as Web Services using technologies that provide remote procedure call interfaces delivering applications over the Web.

On a local scale, resources do not have well-known locations. Techniques such as offering indexes to lists of available resources and lightweight, localised distributed search enable the location of resources to be discovered within peer-to-peer frameworks such as Sun's Project Jxta (Waterhouse, 2001). Introducing additional resource metadata such as summaries and keywords

more readily enables user selection as a manual process. Alternatively, discovery through participation in a link association may suffice to provide access to present resources.

4.6.1.4 Provision Access Control

On the Web, access control to resources is a somewhat straightforward process: site administrators can restrict access based on attributes such as registered user names and the network address of requesting clients. In a localised, pervasive setting, it may not be feasible to administer and manage sets of user names in order to control access to different sets of resources, especially where access includes the ability to modify content.

As with Document resources, it may be inappropriate for any person or service to be able to see the contents of a linkbase, especially if a semantically richer model of linking is employed where annotations regarding the association of two documents or document sections are contained as part of the hyperlink structure. Mechanisms for selectively making subsets of linkbases available for others to use observe similar issues to Internet-scale linkbase publication, although with the added complexities arising due to the transient nature of users and services in a pervasive setting.

4.6.1.5 Provision an Interface

Traditional hypermedia systems often have a simple system interface: they may be a 'black box' that exports a well-defined interface to other processes by speaking a common protocol such as OHP; may offer an open API such as with the DLS; or they might offer an in-line 'transparent' interface, such as DLS's proxy mode.

Being consistent and open is especially important in a system that comprises many heterogeneous services so as to enable interaction. Specific examples of these heterogeneous services within the target scenarios are given in 5.2.6 below.

4.6.1.6 *Provision Linkbase Publication*

Enterprise-level linkbases may be collections of links that have been authored or generated across a domain of resources for an enterprise network, potentially gigabytes in size. Personal linkbases, whilst not likely to be as massive as Enterprise linkbases, can also be large and can encompass a diverse set of topics and themes. In order to exchange link information, if follows that the facility to publish the links and linkbases is required.

4.6.1.7 *Provision Linkbase Fragmentation*

It is unlikely that the exchange of entire linkbases would be desired in the context of the scenarios identified. Therefore, a mechanism for 'fragmenting' linkbases and manipulating those fragments is desired.

Linkbase fragmentation can be achieved by 'anchoring' on partial data, for example, issuing a query to a link service for all link structures with same source, destination, or selection text attributes, in the example of the DLS link model.

4.6.1.8 *Provision Linkbase Fragment Mobility*

Linkbase fragments are not only useful as a tool that enable scoped linkbases for particular environments, but also the ability to move parts of linkbases in response to different queries. Should the underlying link service infrastructure support it, computationally expensive linkbase queries and subsequent link resolutions can be moved to services with superior computational power available (e.g. from a PDA to a laptop). This is particularly interesting in the context of networks of devices with widely ranging capabilities.

Fragment mobility also enable services to move linkbase fragments that are queried frequently by a particular service topologically closer to that service so as to minimise network communications and thus offer a mechanism to improve response time.

There is a trade-off between moving queries to the linkbases, linkbases to queries, and working with linkbase fragments. The link service infrastructure may need to support all three modalities as application contexts demand.

4.6.1.9 *Provision minimal Configuration and Administration*

With many of the desktop-based hypermedia systems available, application configuration is a one-time process. This can be as simple as setting a proxy for HTTP traffic on a browser application for Web-based systems, or manipulating a simple GUI tool to select linkbases and pointers to collections of document resource-bases.

In a pervasive setting, where services may only be available sporadically and many interactions between services being spontaneous rather than scheduled, much of the configuration process needs to be automated or at least distributed out in a secure yet accessible way.

Certain interaction and configuration characteristics may be declaratively shaped by high-level user preferences such as directives that preclude interaction with certain types of service or resource. The granularity at which this declarative configuration is realised will have a large impact on usability in pervasive applications, where such configurations are ephemeral.

Likewise, the administration of the sets of resources that are made available to others, setting up access control lists and trust associations with other people and their services could quickly become too complicated for users to master with any confidence. This complexity is especially important in pervasive devices that may not have tangible user interfaces, for example gesture-based devices.

4.6.1.10 *Provision Persistence*

The need to utilise *local* resources for the dissemination of *local* information and the acceptance of the above issues concerning naming, versioning and local representation means that the hyperstructure realisable by different applications depends on what resources are present at any given time.

As more resources become available, it follows that the hyperstructure complexity increases; more links to more resources are available for inclusion as a result of some query of a link service. However, devices may be equally as likely to leave the scenario as join it, and thus resources and services may go away without notice, for example, as someone disconnects their laptop.

Mechanisms are required that provide a degree of persistence such that, once hyperstructure is generated as a result of resources becoming available or new associations authored, the participants may store relevant fragments of it for later use. For example, when connected to the global Internet and so able to navigate global resources, or so to enable reflection on resources generated during the scenario once absent.

Transclusion

The traditional modality of rendering link anchors in Web-based hypermedia applications has been to underline and colour blue the text that represents the link source anchor, or to have a separate window for "available links". An alternative hyperstructure traversal technique is the concept of *transclusion* (due to Nelson): including in-place the contents of the target of a link at the point in a rendered document where the source anchor was traversed.

These renderings result in new resources that, even if hidden from presentation, may contain references back to their original sources. Come read-time for the user, those resources may no longer be available. It may be appropriate to capture the 'new' rendered document as an alternative to attempting to capture the entire context necessary to recreate the hyperstructure realised in, for example, a contextual link service, as discussed above.

4.7 Summary

Hypermedia linking has traditionally been seen as purely an electronic medium and much of the body of research behind the field almost takes for granted a world grounded in documents, nodes and anchors. Whilst the primary focus of this research is to consider mechanisms that allow links in

the electronic medium to be discoverable, deliverable and navigable when brought together by coincidence, the research also considers aspects of how hypermedia linking maps from the digital into the physical world and can be used to bridge between them.

The three scenarios considered in this chapter are not an exhaustive detailing of those considered in the research of this thesis, but are the three that best capture the necessary detail to support the architecture experiments and typical planned utilisation within scope. The others that were partially explored when scoping this research have been omitted on grounds of brevity. However, one in particular merits mention here as discussion suggests that it challenges the assumption made in Section 4.5.6.

"Cows in a Field" is a scenario similar to the Corridor, above, except here there is no infrastructure networking provision whatsoever, and the only resources and services available are those that are carried 'into the field' by the participants. The challenge to the infrastructure assumption regards the underlying network.

In the Cows in a Field scenario, an ad hoc network is formed between participant nodes that might not be a flat IP-based network. For the purposes of the adoption of our infrastructure, though, it is assumed that either a layer 2 ad hoc networking stack is employed that appears as a flat IP network for the purposes of our infrastructure, or that such signalling exists in the layer 3 ad hoc routing protocol implementation so that the necessary infrastructure components (e.g. *ResourceBases* – see section 5.2.6.1) survive the temporary loss of presence due to networking.

The other implication of Cows in a Field in particular is the new opportunity to introduce technologies such as the Global Positioning System (GPS) as a physical anchoring technique and the additional complexities that such introduction might add to the system.

Where physical tokens such as barcodes and RFID tags are quite concrete – they are sensed and their value the same for the same physical instance – the

use of a physical location cued by GPS is less straightforward. The accuracy of the location sensing needs to be considered such that the process that resolves the sensed reading to a labelled location needs to be sensitive to error; the act of resolving a reading to a label becomes more involved than straightforward table lookup.

However, the treatment of GPS as a physical anchor can be achieved treating the creation, management and resolution processes as another type of *PhysicalTransmogrifier* utility (section 5.2.6.2).

This chapter began by identifying target scenarios in which it is felt that open hypermedia techniques could provide benefit. Then it identified and discussed issues pertaining physical information resources and how they might be integrated into the hyperstructure.

Having identified common requirements and on-going work defining solutions for some aspects of those requirements, we have stated assumptions about the target network environment and how they relate to the nature of the scenarios chosen. That, then, provided the basis for the set of issues and requirements discussed regarding the software infrastructure – the hypermedia middleware – that forms the focus of the rest of this thesis.

The following chapter documents a series of experiments that consider the different architectural requirements of link services as might be applied in these scenarios, informing other infrastructure research projects in which the author has been participative.

Chapter 5 The pDLS Framework

The previous chapter defined a set of application scenarios from which we derived a set of middleware requirements, discussed how physical artefacts might be included in the digital information space, and detailed various activities regarding the 'lower layer' issues regarding infrastructures for hypermedia link services that we wish to develop in such environments.

The primary goal of this research is the enabling infrastructure that provisions hypermedia functionality over local resources, spontaneously available. Having identified in previous chapters that the DLS approach to open linking is a suitable starting point, this chapter begins by considering that architecture and how it fails to meet the requirements in the context of our target scenarios, and how it might then be adapted to do so as a primary participant of a framework of cooperative components.

5.1 Considering the DLS

This section considers the existing DLS when utilised as an in-transport Web proxy through which all client Web resource transactions are mediated.

5.1.1 Interaction / User Experience

In typical deployments, the DLS is an enterprise-level service, deployed and administered by the hosting organisation (e.g. School or Business systems support team), used by many users coincidentally.

In terms of a user experience, the user's system is configured such that any Web-based resource request is mediated through an instance of the DLS proxy, whether explicitly by virtue of navigation in a Web browser, or implicitly as part of an action within applications such as Microsoft Word.

As a result, all Inter- and intranet Web traffic is routed through the DLS, with the result that all content being communicated can be enriched or adapted transparently to the users.

In this use model, all of the resources are (within the context of the enterprise) globally navigable. The linkbases available are thus shared across the entire enterprise, and are typically static and monolithic in nature.

The DLS service can be configured directly by the user, through the use of Web-based forms, with the typical configuration actions comprising resolver (and thus linkbase) selection.

From the perspective of the users within the enterprise, the architecture appears as shown in Figure 2-4 on page 41.

5.1.2 Link Model

Early DLS implementations, like Microcosm before them, used a rudimentary but sufficiently rich description for links in linkbases. Not quite an XML in that there were often no closing tags to complete element descriptions, not that XML was a proposed standard until 1997 (Bray, 1997), the mark up facilitated various metadata in addition to the fundamental requirements of an open link, that is the ability to associate *n* anchors in a link.

(Carr, 1995) gives the following example of a link from an early DLS linkbase:

```
<link type=local>
      <src><doc>http://diana.ecs.soton.ac.uk/~lac/cv.html
            <offset>
            <sel>Microcosm
      <dest><doc>http://bedrock.ecs.soton.ac.uk/
            <offset>
            <sel>The Microcosm Home Page
<owner>Les@holly
<time-stamp>Fri Mar 31 13:32:34 GMT 1995
<title>Hypermedia Research at the University of Southampton
```

Figure 5-1: An example DLS link from (Carr, 1995)

DLS implementations prototyped for the research investigating link service architectures updates this linkbase model so that the linkbases are well-formed XML, the timestamps ISO8601-compliant (Visser, 2000), and the owner attributes Universal Resource Identifier fragments which, for the purposes of the processes using the framework, can be treated as opaque symbols (Berners-Lee, 1998).

```
<link>
  <src>
    <resource>http://diana.ecs.soton.ac.uk/~lac/cv.html</resource>
    <offset/>
    <term>Microcosm</term>
  </src>
  <dest>
    <resource>http://bedrock.ecs.soton.ac.uk/</resource>
    <offset/>
    <term>The Microcosm Home Page</term>
  </dest>
  <owner>http://holly.local/concepts#lac </owner>
  <time-stamp>2000-06-04T04:02:29+0100</time-stamp>
  <title>Hypermedia Research at the University of Southampton</title>
</link>
```

Figure 5-2: Adopted Link model

The explicit attribute specification of link type has been removed because the type of the link can be computed by the presence (or, more typically, lack) of CDATA sections from the link anchor elements. In the case of the example above, this is a 'local link' in Microcosm terminology because it creates an association between a term at any location in a source resource to a specific anchor (in the case, the entire document) in a remote resource.

5.1.3 Query Model

There are four core approaches to generating hyperstructure as a result of analysing different resources queried of the DLS. These are:

1. Query on all resource data

In this strategy, the resource for which links are sought is analysed in its entirety. In the case of text documents, this may involve the process of looking up each word in the configured linkbases; it may involve combinations of words or even sentences.

In the case of other media types, e.g. image data or music, it involves bespoke feature analysis processes on the entire resource, which are then matched against linkbases. In the case of images, image analysis techniques such as shape and texture recognition can be employed to extract symbolic features, for example 'car' by analysis of shape, or 'water colour painting' by analysis of texture and colour (Lewis, 1995), or of the melodic pitch contour of a musical piece (De Roure, 2000a).

2. Query on subset of resource data

The explosive nature of querying linkbases with every single word of a textual resource can be limited somewhat by the adoption of 'stop words' - words or phrases that, when observed, are not queried against for they are sufficiently common as not to be likely to provide interesting or relevant results. This includes words such as 'the', 'of', and 'and'.

Further, various optimisations exist that reduce the query space for a number of the strategies above. These include word stemming, where tense and gender suffixes are removed from words to simplify the query made (e.g. a query on 'removing', 'removed' or 'removes' all result in the same query for 'remove').

3. Query on derivatives of resource data

Further to stemming, the identified terms can be referenced against thesauri that can provide more or less general terms. An example of this approach in use is in the COHSE (Conceptual Open Hypermedia Services Environment) project. COHSE comprised an ontology service that facilitated cataloguing of

documents and an adapted DLS that enabled resources to be linked based on the same or similar concept descriptors (Goble, 1999).

4. Query on resource metadata

Where the above strategies involve the analysis of potentially large volumes of data, a different approach is to query for links regarding a resource based on the metadata available regarding the resource. Such metadata typically includes the resource's name or location (e.g. URL); author-provided metadata; or generated metadata, for example, a thematic summariser process that generates the *n*-most important terms of the document, defining its 'context'.

This approach treats all media forms as equal, and does not require much by way of resource analysis.

5.1.4 Example resolutions

This section describes two example link resolutions using the DLS model, with explicit discrete queries of the service.

A query constrained with source document anchor X and source anchor selection text Y should match (and therefore make available) links from the available linkbases with the following constraints:

Specific Link. Source document is X and selection specified is Y
Local Link. Source document is X and no selection specified
Generic Link. Source document is unconstrained, selection specified is Y

Likewise, a query whose only constraint is that the destination anchor is specified as Z should match only those links that share the same destination. This does not cover the entire spectrum of possible link resolutions against a linkbase, but serves to illustrate the different kind of resolution results we expect from a link resolver.

Given the example linkbase as shown in Figure 5-3, a query on the term *hypertext* in the context of the on-line course notes for the course CM316 would match link `12` on the basis of matching an unconstrained source resource selector but matching selection text. A query on the term *multimedia* on the same resource would result in both `11` and `12` being made available.

```
<linkbase title="example linkbase">
<!-- A local link -->
<link id="11">
<src>
<resource>https://secure.ecs.soton.ac.uk/notes/cm316/</resource>
<offset/>
<sel>multimedia</sel>
</src>
<dest>
<resource>http://dictionary.oed.com/cgi/entry/00318167</resource>
<offset/>
<sel/>
</dest>
<owner>http://bill.local/concepts#mkt</owner>
<time-stamp>1995-02-06T14:42:21+0000</time-stamp>
<title>Multimedia Definition</title>
</link>
<!-- A generic link -->
<link id="12">
<src>
<resource/>
<offset/>
<sel>hypertext</sel>
</src>
<dest>
<resource>http://www.hypertextkitchen.com/</resource>
<offset/>
<sel/>
</dest>
<owner>http://bill.local/concepts#mkt</owner>
<time-stamp>2002-08-02T03:11:01+0100</time-stamp>
<title>Hypertext Resources</title>
</link>
</linkbase>
```

Figure 5-3: Example linkbase

Note that the presentation of these links to the user can be manifested in numerous ways. The implementation by Carr as presented in (Carr, 1995) offers a user-configurable suite of rendering options, including familiar 'underline and colour blue' on anchor features (which can be restrictive should more than one link match a source anchor), annotations (e.g. an asterisk or a bibliographic-style decoration) which are themselves 'href' anchors in the HTML, or as a set of available links presented as a separate

Web page comprising a list of titled HTML anchors, separate to the content of the resource being linked from.

5.1.5 Observations

Using a single enterprise-level DLS instance for all users has been successful with regards provisioning hyperstructure across globally navigable resources, utilising resolvers and linkbases local to the single DLS process instance, for example in the MEMOIR project (De Roure, 1998). Users have functionality to generate and then add their own linkbases to the system, which are then shared amongst others within the enterprise, at the expense of being a systems-level administration, i.e. the service needs to be reconfigured and subsequently restarted for the new linkbases to become available.

Link resolution is performed at the proxy component, although as discussed earlier, work on distributing the query to multiple, distributed, resolvers has been exercised in Internet-scale systems.

Considering the target scenarios of this thesis, the architecture and service characteristics of the DLS are unsuitable. Firstly, the resources (both documents and linkbases) we wish to communicate are coincidentally local, and not globally navigable. Also, whereas it may be permissible for every member within an enterprise to be 'trusted' to the extent that all resources, linkbases and resolution services are freely available, that is not the case in our scenarios.

5.2 pDLS Framework

To achieve a satisfactory DLS-based architecture in light of the observations and requirements of the previous chapters, the proposed usage modality for our 'pervasive information fabric'-compliant (due to Weiser, as on page 49) DLS, hereafter the pDLS, would be as follows.

5.2.1 User Experience

Before the event, e.g. in the case of the Meeting Room or Conference scenarios, users would publish the resources that they wish to either make use of, or make available for others. Ideally, this would be an activity of a fully integrated content management system that is distributed across the entire computing landscape that the user interacts with.

For the purposes of developing the pDLS, this involves the use of a ResourceBase utility that, amongst other things, copies instances of resources to share during the scenario to a bespoke store, similar to a 'My Shared Documents' folder on a Windows Personal Computer, or 'Shared Items' category on a PocketPC.

Users also publish the linkbases (or subsets thereof) that they wish to both use themselves, or make available for other participants' resolvers to query when in-scenario.

Where the primary purpose of the pDLS is to generate hyperstructure across coincident resources, linkbases and link resolvers, discussion in section 4.6.1 above detailed additional tasks of our information fabric that suggest the presence of additional utilities, such as resource transcoders, thematic summarisers, physical anchor transmogrifiers etc.

The pDLS comprises a framework of components, whose core consists of the hypermedia link service with functionality grounded on the experiences with the traditional DLS. In the expected use, the pDLS framework of components would be packaged for the various devices that the user intends to deploy in the scenarios.

5.2.1.1 Scenario time

The desire is for minimal configuration and administration, and therefore the facility to simply 'launch' a single component that then marshals the framework's bootstrap, enabling the user to interact with other local participants.

This requires that the various infrastructure components self-configure, discover other local services, then authenticate, register, and connect with them as appropriate.

The principal use interaction with the pDLS should be as familiar as possible for user's typical information interaction, and therefore the modality is that of the pDLS link service being interfaced with as a Web-browser proxy service.

Explicit user interaction with the pDLS for the purposes of service configuration should be through a web-based form. Whereas the DLS configuration interface presents options for link presentation and linkbase selection, the desired explicit interactions with the pDLS include:

- Local linkbase actions: activation as a resolution resource, publication
- Remote linkbase/resolver actions: activation as a resolution resource
- Link compilation for explicitly identified resource
- Monitor information space events (e.g. new resource or service announcements)
- Interact with information space utilities (e.g. add a transcoding component to the HTTP transport path, use an information space browsing tool to browse coincident resource bases)

5.2.1.2 Deconstruction

As participants leave the scenario, there is a requirement for the information space to be deconstructed, with resources, utilities and services no longer being available for use by other participants.

Ideally, the pDLS infrastructure would announce the departure of entities from the information space so that other participants can react (e.g. issue one last request, enact a 'last will and testament' service interaction). However, such functionality should not be assumed so that a participant failure case (e.g. unanticipated loss of connectivity or power) does not render the entire information space unusable for the surviving participants.

Other, still live, participants should survive the departure, with the appropriate utilities and services remaining still being serviceable within the infrastructure.

5.2.1.3 Post-event

From the perspective of the absent user, the pDLS might still function as pervasive information fabric of one participant, i.e. only using local resources, local link services, resolver and utilities. This assumes that the pDLS infrastructure is self-contained and componentised to the extent that the smallest pDLS is the unity pDLS.

Additionally, the utilities for Resource and linkbase publication should be such that it is possible to extract resources and linkbases for interrogation and use without the rest of the pDLS framework active.

5.2.2 Interaction

Web-based delivery of the service-controlling graphical user interface (GUI) is familiar to other applications, including the previous versions of the DLS, and therefore is a suitable and readily available approach for the pDLS and its supporting infrastructure.

Based on the previous work, the observations of the scenarios and their arising requirements, the pDLS GUI is as shown in Figure 5-4.

The user interface is divided into four main functions:

1. Resolver management. Familiar from previous DLSes, this panel is used to configure which linkbases are 'active' for resolvers.
2. Utility command and control. Where utility processes such as transcoding proxies, resource caches etc. are available for interaction, a handle (HTML button) to their controlling interface is presented in this panel
3. Infrastructure Event viewer/response. Partially for debug purposes and partially as an aidance for monitoring actions pertaining the

various entities in the information space, the event viewer panel provides a look-through into some of the interactions between components in the infrastructure, in a human-digestible form

4. Resolution results. The most important panel of the pDLS interface is the results panel in which resolution are presented as lists of available links pertaining the last resource navigated by the client's web browser (assuming that the navigation wasn't to the pDLS service interface)

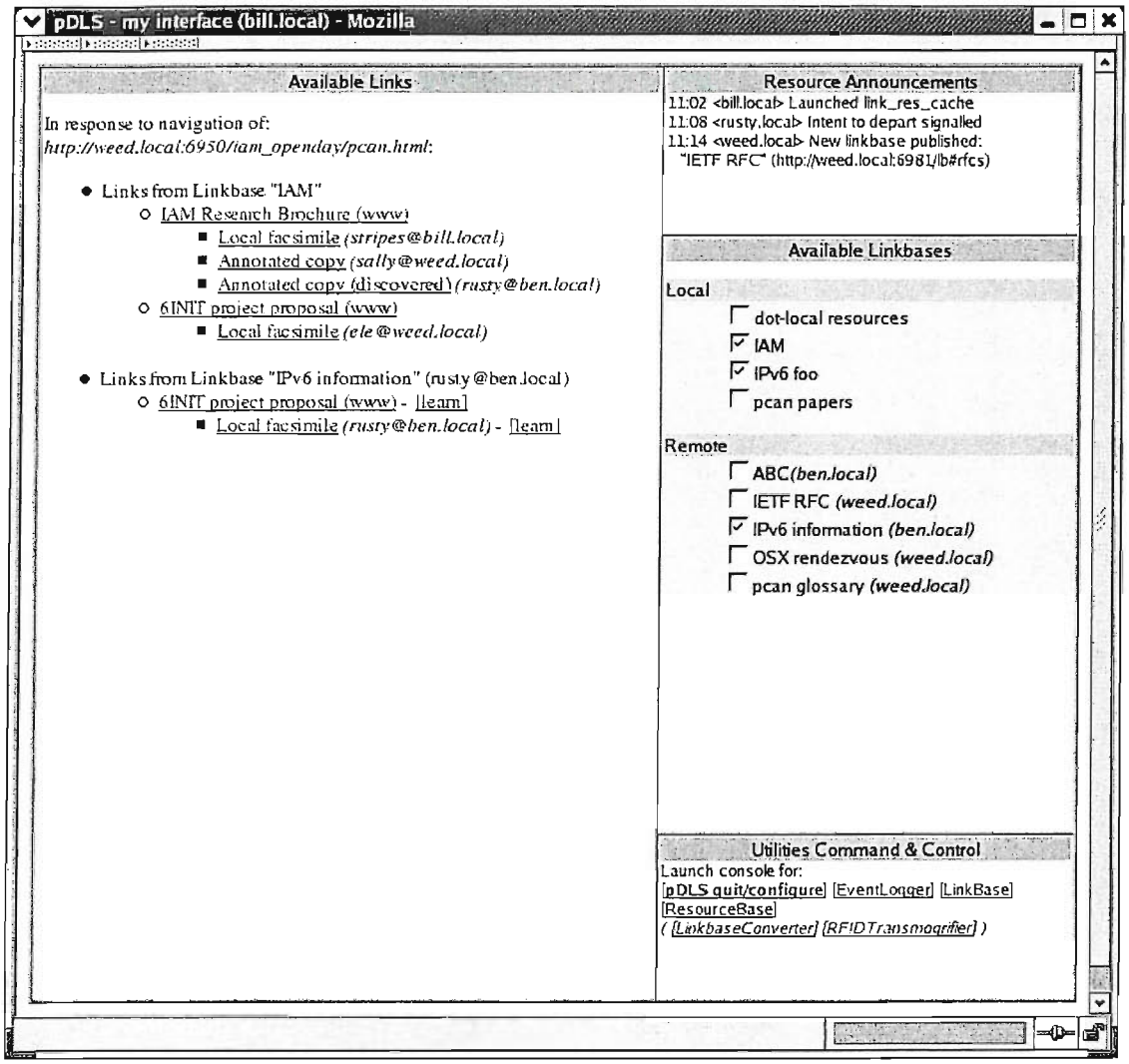

Figure 5-4: pDLS User Interface

5.2.2.1 *Resolution modality*

One link resolution modality is for the proxy component to compile the set of links and their anchor points within a resource whilst the resource is in transit through the proxy component. There is a readily identifiable operational latency with this approach that is greatly exaggerated when the resolver processes are remote to the proxy, even when on the same machine.

An inefficient resolution strategy that performs queries on every term (word) or compounds of words (two, or three at a time) can cause an explosion in the amount of queries performed and the amount of data passed between resolution processes, especially as the size of the resources grow, and as the number of resolution processes increase also.

Where techniques such as word stemming and partitioned resource querying serve to reduce this burden somewhat, the perceived latency in final resource delivery to the user is often unacceptable.

An alternative approach to in situ source anchor and Web link injection on resolved links is to provide a contextualised result set of applicable links, asynchronous to the original navigation query. In this modality, the requested resource is delivered unaltered to the requesting client, and a copy of the resource held within the link service for asynchronous link compilation. As link resolution events occur from whatever resolver processes have been invoked, their results can be presented (with associated context cues) in a separate window.

Additionally, once the user is satisfied that sufficient links have been resolved, they may wish to see a version of the resource augmented with the additional hyperstructure available in-place, achieving the same view that they would otherwise have seen had the link resolution occurred in synchrony with the resource navigation query, but without the (potentially crippling) latency.

5.2.3 Interface architecture

The client interface to the pDLS remains familiar: a single interface to the entire information space realised as a component that exports both a web server interface for queries of the system, and a web proxy interface enabling in-line analysis and retrieval of content. In the case of the pDLS, this content is sourced from both the global Internet (should there be such a connection to the 'outside world'), and from other HTTP-speaking resource bases on locally coincident participants.

Whilst the client interface to the information space clearly resembles its DLS ancestor, the internals of the pDLS are substantially different so as to cater for the spontaneity and ad hoc-ness in light of the challenges identified in previous chapters.

The roles of the pDLS core components include:

- Instigate queries of link resolver processes dependent on configured linkbases
- Maintain metadata regarding resources 'seen' as available
- Retrieve resources requested by user (i.e. function as a web proxy)
- Provide interface to utilities that manipulate resources and infrastructure events including linkbase or resource publication, successful link resolution, and alternate versions or representations.

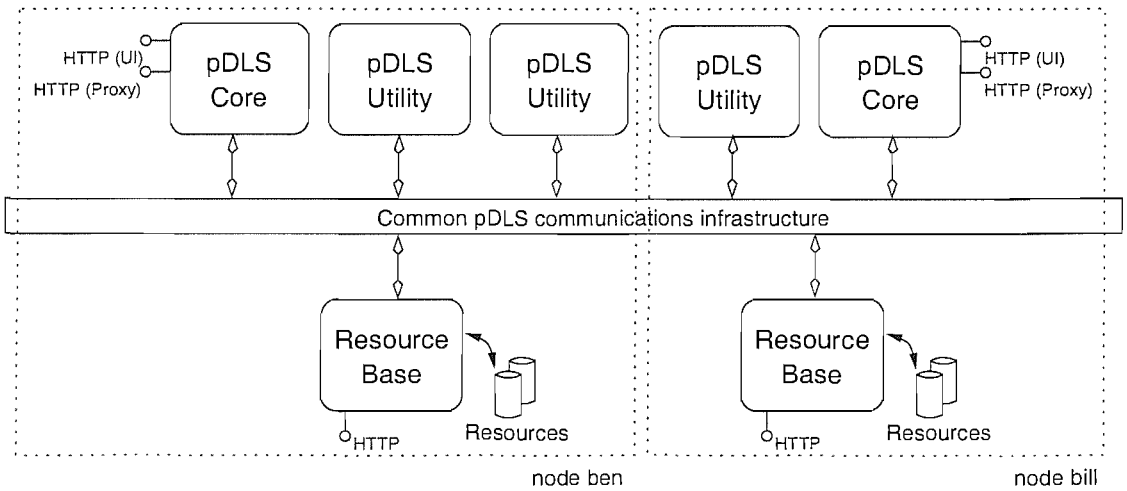

Figure 5-5: pDLS architecture

Essentially, the underlying infrastructure should provide a communications substrate that is shared between all of the components in a manner that is as transparent as possible to the user, retaining the familiarity of the tools that they would use in their normal work environment, except where the additional utilities explicitly require otherwise. That is, it should behave like the Web, but for local people with local goals.

5.2.4 Linking concepts

When considering the nature of the links in each of the scenarios, there is scope for confusion as regards the description of the concept being anchored upon.

In the case of traditional hypermedia links, particularly with the DLS, the referent is typically a well-qualified resource; for example a URL comprising a descriptive access protocol, information regarding where the resource is served from, path information, parameter information, and even dynamic query information – precisely specifying in a well understood manner how that resource being referenced can be retrieved. Additionally, there may be other declarative data such as offset into resource, selection feature, etc.

However, when non-document based concepts are to be captured in associations, the nature of the encoding has not been standardised, and a number of different possible solutions exist.

For example, an anchor on the concept of Mark Thompson the person could be encoded in a number of different ways:

- Using Web-accessible, well-formed URLs as unique identifiers, the content at which contains metadata about the concept being anchored upon. e.g. `http://www.ecs.soton.ac.uk/info/people/mkt`, which contains human-readable descriptive contact and ancillary information about the person

- Using opaque Uniform Resource Names (URNs) (Moats, 1997) as tokens that serve as a single common identity designating equality for all associations that reference it, although not necessarily having any means to resolve to any other form (human-readable or otherwise). e.g.
  `urn:slf:mark.k.thompson`

- Using URI fragment identifiers that, when resolved, do not serve valid content, but act as (structurally significant) tokens within the domain of a knowledge system in that there are processes that can translate the token into other machine-understandable forms. e.g.
  `http://www.ecs.soton.ac.uk/info#person-02172`

- Using some other resolvable token, e.g. LDAP directory service URL that, when resolved, would provide metadata regarding the concept including, perhaps, other tokens by which the same concept has been described. e.g.
  `ldap://ldap.ecs.soton.ac.uk/ou=ECSUsers??sub?(uid=mkt)`

Each of these approaches provides a different digital representation of the same physical concept, as they could equally a digital concept. Some of them can be resolved or navigated to retrieve declarative information regarding the concept they label, others merely opaque tokens. It is unlikely that associations authored referencing one representation for a concept will also have been authored representing the other possible forms, whether due the fact that the author was not aware of the other forms, or because there are so many of them.

Therefore, in order to ensure that all intended associations are served when querying on a particular concept's participation, there is a need to support alternative representations of the same concept. This can be achieved in a number of different ways, including:

1. Use of a resolution service that 'folds' all known equivalent concept anchors together as a pre-filter on linkbases. This has the advantage of introducing a system-wide uniform conceptual representation, but at the (potentially great) expense of modifying all of the linkbases present

– a task that would need to be repeated each time a new equivalent representation was discovered;

2. Explode every occurrence of an association that references one representation of a concept with new associations that reference all other known representations. This requires stateful knowledge of all concept mappings, and might introduce redundant replication at the expense of significant computation time;

3. Maintain an additional linkbase of associations that generate 'equivalence' links between different anchors that are observed to anchor the same concept. Whilst this leaves the original links untouched, it does incur additional load for the query resolver in that the link matching process now has to consider the combination of the query with each of the members of the set of equivalent concepts when filtering links to return;

4. Do nothing, and assume that a third party process folds equivalent concepts to the same anchor.

The most attractive of these approaches as far as a generic evolution of existing link services is concerned is the last one. The specifics of how individual link resolvers cater for conceptually equivalent anchors in instances where links do not consistently reference them is beyond the scope of this work, although it is closely related to the area of concept folding when mapping between different ontologies of knowledge structure (Kalfoglou, 2003).

The third approach described above fits with the model proposed as a solution to the Versioning and Representation requirement, discussed in 5.2.7.2 below.

5.2.5 Resolvers v. Linkbases

The initial approach within this framework has been that the URL describing the location of a linkbase implicitly identifies the resolver that is capable of

querying it; that the linkbase URL is not an opaque token, but has location semantics.

For example, the linkbase identified in the GUI example above, `http://ben.local:6970/lb#ipv6info`, serves as an identifier for the 'IPv6 Info' linkbase on node `ben.local` within the local network. The linkbase URL also provides a cue to the local pDLS Query Dispatcher (the sub-process that enacts linkbase queries as a result of a user's or process's request) as to the location of the resolver process that can query the identified linkbase for results.

This notion is explored further in the next chapter, where different internal architectures for the pDLS component are considered.

5.2.5.1 Linkbase Types

A familiar behaviour for link server systems is for link resolution queries to be matched against a set of pre-authored links residing in sets of linkbases. Within the pDLS, there are numerous other types of Link Resolver, some of which are dynamic in nature. Four examples are presented here:

Resource base metadata

An approach to enabling navigability to resources that are local representations of resources that might be targets of pre-authored links in user's linkbases when in their globally addressable ('home') location is for a resolver process to query a locally-maintained map of resource metadata.

This can be realised in one of two ways. It can be a process of link composition, where the global URL is returned in the result set of other resolutions the local pDLS automatically rewrites the result to reference the local copy. Alternatively, where resolved links identify non-local resources, the local resolver could then issue a query against the resource base metadata resolver to attempt to discover local representations.

The former lowers the fidelity of the original resolved link, and the user would lose the ability to save that link for later use (e.g. when connected to

the Internet). The latter risks an explosion in the number of available links returned as the result of a query.

Keyword extraction

If resources already have keywords identified, either through available metadata (e.g. explicit mark up in HTML documents) or by performing keyword extraction on the fly, these keywords can then be used as a means to generate generic links in a local linkbase that remains relevant to local resources for the duration of the scenario. This enables new resources brought into or generated during the scenario to immediately become a navigable destination within the hyperstructure.

Anchor manipulation

Another source of links in the case of HTML or other embedded-anchor resources is the hyperstructure that is already present. Identifying existing anchors in the resources and extracting their anchor text and destination into a live linkbase enables both generic link and specific link creation: the link from the local resource anchored on the anchor text to the identified global resource, and the link on the anchor text generically to the global destination.

5.2.6 Role of Utilities

The core functionality of the pDLS is to provide access – enriched with local generated and pre-authored hyperstructure – to local resources coincident on the local network. The components that enable that are the controlling GUI, the (HTTP) Transport Proxy and its constituent components, later described as Query Dispatcher and Link Resolvers.

Every other function is considered a 'utility', with the collection of utilities and pDLS core components comprising the pDLS framework. Utilities may generate or respond to framework events (e.g. link resolution, resource publication); they may manipulate resources directly (e.g. publication, thematic summarisation, linkbase manipulation); indirect manipulation (e.g. transcoding such as in section 3.2); or simply provide administrative functionality (e.g. framework monitor).

Some of the utilities of particular interest are described below:

5.2.6.1 *ResourceBase*

The ResourceBase utility is the process through which all resources are delivered to requesting clients. In many respects, the ResourceBase can be considered a fully-fledged Web server and instances of the ResourceBase can be navigated with any HTTP-compliant client (i.e. not solely the pDLS core components).

The ResourceBase process also maintains mappings from local resource URLs to alternate versions or representations of the same resource, some of which may be global. This resource metadata is exposed as a linkbase that can be, with the coordination of an appropriate link resolver component, included in the search space of the pDLS such that users can be made aware of the alternative instances when other instances are navigated as resources, or queried as part of an iterative link query (see section 5.2.7.2 below).

5.2.6.2 *PhysicalTransmogrifier*

The observation made in section 3.3.2.3 was that the participation of physical artefacts in the hyperstructure could be facilitated by an appropriate anchor resolution process; that the physical anchor could have a digital representation or 'proxy' in the information system, and therefore be participative in the scenario.

The PhysicalTransmogrifier utility is the mechanism that resolves physical anchors (e.g. scanned-barcodes, detected RFID tags, iButton docks) to a locally registered digital resource representing the anchored object such that it can participate in the local hyperstructure.

Different classes of PhysicalTransmogrifier should exist for the different types of physical anchor, and their deployment comprises a binding phase, where the physical token is sensed and bound to a resource in a local resource-base. The perceived meaning of the anchor is very much a 'tag' of a physical object, with no additional semantic meaning captured.

This follows a similar pattern to Kindberg's Pulp Computing (Kindberg, 2003), but rather than a centralised anchor resolution registry, the binding between physical tag and local hyperstructure anchor is maintained within the local scenario. The binding could persist and be realised as a local cache of a globally asserted binding as part of a Pulp Computing-alike system.

5.2.6.3 Third Party data and LinkbaseConvert

The Link Model adopted for the pDLS as discussed in section 5.1.2 above is a natural evolution of the link model employed by the original DLS.

One level of interoperability with other link server systems is enabled by the specification of a clean Link Resolver interface: legacy open hypermedia systems can be integrated with the pDLS through the adoption of a shim component that implement the Link Resolver interface and mediate link resolution queries issued by the pDLS Query Dispatcher to the legacy system's query interface.

Whilst this protocol translation enables a degree of extensibility, it does introduce additional issues, such as how access control or other utility interactions are adapted to suit the functionality of the integrated system.

A second approach to interoperability is to adapt the link resolver component of the pDLS core to parse the contents of foreign linkbases and interrogate them in response to queries.

This approach requires that the link model of the foreign linkbase can be translated to fit the pDLS model, and that the nature of linkbase querying fits with the pattern-matching term and anchor based modality that the pDLS employs.

A different approach to integration of third party systems is to convert their source data (their linkbases) to the adopted pDLS link model, and then treat those as native as resource data for link resolvers in the pDLS core. A utility (separate from the framework) for this conversion from a variety of different

linkbase formats, including the original DLS model and a rudimentary level of support for XLink (De Rose, 2001).

5.2.6.4 *LinkbaseFragmentation*

Using a combination of the techniques discussed in section 4.6.1.7 above, the LinkbaseFragmentation utility provides mechanisms for creating new linkbases as a result of querying remote linkbases through the pDLS, or by direct manipulation of specified (navigable) pDLS-format linkbase data files.

These fragments can then be published as linkbases for query within the framework, or 'externalised' for consumption by, for example, someone else within the scenario as a new resource for integration to a link service elsewhere.

5.2.6.5 *ChumpBotLinkBase*

The IRC protocol is discussed as a candidate communications infrastructure in section 6.1.3.1. Other uses of IRC for hypermedia applications (if not explicitly termed as such) include linkbase authoring as a side effect of the more typical use of the protocol: human-human conversation.

An example of this is at academic conferences where wireless networks are becoming prevalent, and services such as on-line proceedings, demonstrations, and collaboration services such as IRC and, offering more persistence, Wikis (Cunningham, 2003). Also, people unable to attend the event can join the attendees virtually, discussing the papers, suggesting questions, etc.

Ignoring for a moment the argument that links can be considered as simply another form of annotation and therefore any recorded transcript of a dialogue on IRC pertaining to physical events being considered a 'linkbase', bots such as the Daily Chump Bot[6] have also been used to author more traditional hypermedia links. The context of these links is anchored implicitly by the various features of the medium, such as the person creating the link,

---

[6] Daily Chump: http://usefulinc.com/chump/

the channel of IRC in which the comment that created the link was conveyed, and explicitly by any additional data provided at the point of creation.

A trivial extension of bots like Daily Chump enables the creation of linkbases in the format required for integration into the pDLS as link resolver components.

5.2.6.6 FrameworkBrowser

The FrameworkBrowser utility supports the Infrastructure Event Viewer/Response panel of the GUI (Figure 5-4 above) by offering an interface for users to maintain a live view of the various pDLS framework components that are present (and live) in the network.

Instances of utilities can be discovered using the adopted mDNS-SD approach as discussed in section 4.5.5.1 above, or as a result of their presence announcement when bootstrapped (e.g. by a node joining the scenario), depending on the communications substrate employed by the pDLS framework.

Once discovered, the FrameworkBrowser component offers additional information regarding each component discovered, e.g. name, type, and any other relevant data depending on the type of the component.

5.2.6.7 EventLogger

The EventLogger is an example of a debugging utility that adds no functionality for users of the system, nor does it enrich the space in any way. Rather, it offers a more detailed window onto the interactions between components (depending on the choice of underlying pDLS infrastructure) as a mechanism for debugging, and to satisfy pure curiosity about the activity of individual processes.

5.2.7 Requirements Evaluation

The exploration of different architectures for the pDLS core and the common communications infrastructure forms the focus of the next chapter. Reflecting

the framework of components approach proposed against the requirements raised by the previous chapter leads to the following observations:

5.2.7.1 *Local Naming*

The IETF zeroconf working group interest in un-administered spontaneous networking has resulted in the development of node name resolution services that do not require any preconfigured infrastructure services.

With the mapping of automatically configured 'link local' IP address to 'link local' navigable node names maintained by domain name services (mDNS) present on each provisioned node, it is possible for local resources to be addressed by URLs that only have local significance, and are navigable using existing tools.

That is, a device whose DNS resolver library is mDNS-aware will, when tasked by the client's interface application, first query local mDNS responders for the address of a node named in a URL before attempting to connect and retrieve the resource. A key feature – and in this circumstance, benefit – of the proxy approach is that the name resolution process is enacted at the proxy, not at the browser.

As a result, it is only the pDLS proxy component that has to be mDNS-compliant. This has the benefit that, given the experimental stage of development of the zeroconf approach to local services and the accompanying lack of availability of system-provisioned resolver tools, the functionality can be implemented entirely in the pDLS components, without relying on underlying operating system support.

5.2.7.2 *Versioning and Representation*

The pDLS is naming agnostic in that the purpose of the transport proxy component is to retrieve the resource identified by the URL provided by the client, communicating associated query data and HTTP headers as appropriate, and deliver the retrieved content both to the client and to any configured link resolvers and utilities.

However, there are cases with participants bringing pre-authored linkbases and copies of resources from the global Internet into the local information space where navigation and retrieval are required. A solution within the proposed framework has been to provide a URL mapping service that maintains data about the resources provided by a participant that comprises:

- Global identifier. *The URL of the resource in its usual state, e.g.* `http://w3.org/TR/2002/REC-xhtml1-20020801`
- Local identifier. *The URL of the resource locally, e.g.* `http://ben.local:6970/w3c.org/TR/2002/REC-xhtml1-20020801`
- State. *Whether the resource is an exact facsimile, a revised version, or a representative (e.g. possibly transcoded) copy*
- Owner. *The identity of the owner of the resource with regards the local scenario, e.g.* `ele@ben.local`
- Comment. *Free-form comment regarding the resource, for the purposes of local information space browsing utilities*

This fits the link model specified for the pDLS, where the State and Comment elements are combined to map to a link description. This suggests that resource metadata can be considered as link data.

One approach to provisioning versioning and representation within the pDLS has been for a dynamic linkbase of these resource description tuples to be maintained and thus matched as part of the link resolution process where selected as an enabled linkbase by the user.

5.2.7.3 Discovery

The framework approach provides four mechanisms for resource discovery within the local information space.

Browsing

One interaction that a user can employ with the local space is to navigate the various present resource bases directly, perhaps serendipitously, and discover resources present in the space at that time.

Resource bases can themselves be discovered through the pDLS GUI's framework event pane; through the FrameworkBrowser utility; or by stepping back up from resources discovered within those bases by virtue of another technique below.

Search

Where the ResourceBase components local to each node facilitate HTTP browsing, third party tools such as ht://Dig[7] can be deployed and scheduled to harvest search data over each local resource base. Consideration is also required for indexing resources that are generated or added whilst within the scenario, e.g. captured from a remote participant.

This introduces an overhead in that index generation requires both computational time and disk space to cache the results, and is likely a process best served in the pre-event phase (e.g. when publishing the set of resources predicted as required to the local resource base).

Rather than being a function of the pDLS directly, resource searching is considered a utility supplementary to the infrastructure.

Notification

Events such as resource or linkbase publication being displayed in the UI for the pDLS gives rise to a passive method for discovery based on observed behaviour of other participants.

For example, when a linkbase is marked as being available for remote resolvers to use, other participants in the space can be notified of the linkbases' location (and other metadata, such as title or purpose), and then can opt to add that linkbase to their local pDLS' resolver targets.

Association

Discovery by association is achieved whenever a link is resolved that results in the user being exposed to a new resource, local and therefore navigable or otherwise. An example would be a link in a remote linkbase that resolves to a

[7] ht://Dig: http://htdig.org/

resource on the Web that has no local representation as per Local Naming above. The link discovered might be added to a local linkbase for future use, ditto any local versions of the resources downloaded to the local client, whether implicitly by caching or explicitly using the familiar 'Save As...' functionality of the browser, perhaps then publishing to a local ResourceBase.

Remote resolver processes might announce successful link resolutions to their peers (e.g. source resource, context of link matched, destination resource, linkbase resolved in, etc.) and therefore enable discovery by other participants' resources as a process of another user's navigation within the space. This would be ethically questionable practise in mixed scenarios where, for example, the level of trust between participants might not extend to incidental actions being captured and their effects observable by third parties. For the purposes of an enabling technology, however, the ability to perform such discovery through remote observation is interesting.

5.2.7.4 Access Control

The existing architecture provisions two levels of access control for third party access to local resources and services. They are 'private' where only the resource or service owner has access, and 'public' which are available for all.

In a production environment, the extension to a more granular, group-based access control model would be desirable, at the cost of increased complexity and loading on the user with regards configuration and administration.

The choice to proceed with a restricted access control feature set is the result of a trade-off in the consideration of the underlying pDLS query infrastructure, which is the purpose of the following chapter, and the model of information and services that sit atop of that infrastructure. This work considers enabling infrastructures that does not preclude the notion of access control and other security issues.

5.2.7.5 *Interface*

As discussed above, there is a single client interface to the pDLS that facilitates access to key functionality for the purposes of exploring link service infrastructures within the target scenarios.

Actually, there are two interfaces:

1. The explicit GUI exposed by the user's immediate pDLS service as a series of web pages through which all facets of the infrastructure can be used and managed (enacted as a set of CGI form-based interactions of the pDLS Interface component)

2. The web proxy component through which all external content is routed so as to be enriched with hyperstructure or affected by other intermediary processes where configured, transparently to the user

The nature of the underlying infrastructure implementing the conceptual framework presented above enables third party utility processes to participate in the information space. Whilst their user interface is mediated through the unified GUI presented above, for the purposes of development there are often alternative interfaces available, e.g. as realised through the utility components such as the FrameworkBrowser, above.

5.2.7.6 *Linkbase Publication*

Linkbase Publication can be as trivial as adding the location of a local linkbase in an appropriate form to the local pDLS resolver set (see the example GUI above, and section 5.1.2 above). With consideration to access control as above, published linkbases can be marked as private (i.e. local resolvers only), or publicly available, at which point their presence and location is announced to other discovered parties, and to parties that announce their subsequent arrival to the infrastructure.

Linkbase publication does not imply the ability to browse linkbases independent of resolver processes. It is not intended functionality that an entire linkbase can be readable as an activity in its own right, c.f. queried as

part of link resolution. However, there is nothing preventing utility processes from examining linkbase resources, should the chosen implementation offer such an interface in addition to query. The chosen common link model presented earlier in the chapter is a valid XML, and therefore in the trivial case, an XSL transform (Clark, 1999) can be applied to render the linkbase into something human-readable.

5.2.7.7 Linkbase Fragmentation

Linkbase fragmentation is primarily intended to be a feature of the pre-event phase of the scenario where users publish the sets of links that they might wish to make use of or make available to other participants, likely as a subset of a larger linkbase or linkbase collection.

Fragmentation is very much a utility function that manipulates existing linkbases to generate new ones based on selection criteria such as resource or link theme, linkbases comprising exclusively of generic links (i.e. glossary linkbases), or links only pertaining to resources that have been published to the local resource base with accompanying metadata.

5.2.7.8 Linkbase Fragment Mobility

Within the framework proposed, linkbase fragment mobility is a function of the resolver processes, and typically involves the wholesale transfer (copy) of linkbase fragments to be on the same participant node as a previously remote resolver process, for the purposes of reducing the latency of link resolution.

Where appropriate, linkbase fragment mobility is considered in light of the pDLS prototype implementations in the next chapter.

5.2.7.9 Configuration and Administration

Aside from a familiar look and feel, a fundamental motivation for the single explicit user interface to the pDLS has been such that there is only one entity to configure, and that configuration should be a one-time process.

With the exception of access control as discussed above, the common configuration tasks enabled by the UI are grouped by their expected frequency of use:

Frequent (i.e. throughout scenario event):

- Resource base selection and availability
- Linkbase selection and availability

Rare/one-shot:

- Local node name and properties (c.f. hostname and description in a Microsoft Workgroup), although a reasonable default is provided if the user opts not to explicitly configure
- Utility bootstrap and utilisation where default service sets deemed inappropriate
- Configure operating system to route Web-based traffic through pDLS proxy service

5.2.7.10 Persistence

Persistence of the hyperstructure afforded by the presence of linkbases belonging to other participants can be achieved by the capture of links as they are resolved as the result of queries during the scenario event, a form of implicit capture; by users explicitly marking remotely resolved links as to be added to local linkbases as a function of the user interface; or by the deployment of utilities that capture linkbases or fragments thereof whilst remote participants are present in the infrastructure.

Each of these techniques can be used to create or augment local linkbases that can then be configured for use by the remaining pDLS resolvers, enabling the survival of other participants departing the information space.

Likewise, as resources are discovered by any of the means discussed above, copies can be added to the local system, perhaps even a local ResourceBase for later use through explicit action.

An issue with this approach is that only the resource is captured, not the associated metadata, which describes the fidelity of the resource and also possibly a 'home' URL for the primary instance of the resource. Having stored a copy of the resource locally, a separate explicit action to query the source ResourceBase's accompanying resource metadata linkbase (see section 5.2.7.2 above) would be needed to capture the additional context data, if so required.

Link composition

Considering the resource metadata approach as described above, a new link can be authored and made available that associates the freshly captured (and now local) resource with its source (and possibly absent) instance, which would in turn have a link associating it in its former location with a global URL.

The implication then is whether the resolver should automatically attempt to 'follow links' when discovering links that anchor local resources as destinations. To do so would possibly discover other instances or versions of the resource in the local information space. It would also provide an automatic mechanism that transparently caters for cached or captured instances of resources that other participants brought to the scenario, had been discovered by at least one participant before the originator left, which would otherwise render the resource un-navigable.

This is a dilemma shared with other link service approaches, but one that has somewhat unique requirements here. In an enterprise-scale DLS in an Internet-connected site, it may be desirable – even advantageous – to never actually retrieve resources that are the endpoints of links, but to instead only ever consider (reference) the hyperstructure.

Performing such multi-hop link following does not scale well, although multiple concurrent resolvers may provide an appropriate solution at the expense of 'wasted' computation time when the user's intention was single hop, possibly serendipitous hyperstructure traversal. The fact remains that the multi-hop link composition process would introduce significant

computational complexity for little obvious reward: What additional information might be gleaned that otherwise would not have been from the users explicitly requesting each iteration themselves?

However, in the case of the local information space, the reward is clear: by querying for resources that are representations of other local resources, it is now possible to access information resources that would otherwise be un-navigable due to a single participant (and thus the destination anchor for a whole set of links) leaving the information space.

There is a tension here for resolver processes. When deciding whether a link that matches a query should be delivered, the resolver process may be configured to consider whether the destination anchor is navigable within the current context. That is, if a link references a resource on the Internet where no network connection to the 'net exists in the scenario, the resolver may choose to deliver the link regardless, letting some other process (e.g. the user themselves) discover that the remote resource is unavailable but that an association does exist and therefore can be added to a local linkbase for traversal later.

Alternatively (or even additionally), the resolver could then query the available linkbases for local instances of the global resource by querying for links anchored on the global (un-navigable) resource, as suggested in Versioning and Representation, above. The former link is the more useful for the user away from the scenario in that it concerns the original target resource in its original context, and so its utility not realised until after the scenario has ended, or a connection to the Internet available.

5.3 Summary

This chapter presented the link model and query modality adopted by the proposed approach to achieving a distributed link service within the context of the identified scenarios.

The proposed solution comprises a framework of cooperating components, whose primary goal is to support the functionality of the link service resolving and delivering links between coincident resources in the local network.

Having reflected the framework approach against the identified requirements here, the following chapter documents a series of prototyping experiments investigating different infrastructures for the pDLS, and the common communication infrastructure identified as part of the framework definition proposed by this chapter.

Chapter 6 pDLS Architecture Experiments

The previous chapter identified a candidate approach for enabling the desired interactions as a framework of components that combines a DLS-like approach to hyperstructure realisation within the target scenarios, and a set of utility processes that offer support to the information space. This chapter documents a series of prototype developments that serve to explore different architectures for the pDLS, focused on its core components: the query interface, the transport proxy and the link resolver.

The methodology behind the choice of these experiments has been to first examine the nature of the pDLS transport proxy and whether a decoupled architecture might be of more benefit, and then to consider different technologies for the link resolution processes and their affordances. That is, two streams of experiments, first looking at communication patterns between decoupled and distributed DLSes, and then different computational models for those components co-operating in a distributed environment.

The experiments are non-exhaustive within each stream, the implementations sufficiently developed that the relative merits of the approaches within the context of the research has been observable. Within each stream, the characteristics and observations of each experiment are drawn-through into subsequent experiments, enabling an evolutionary approach to requirement satisfaction that also gives rise to additional elements of interest in later experiments.

With these link service architecture experiments, there are no changes to the interaction model of the clients of the pDLS. Whether querying explicitly for links using the CGI-like interface or using the service in Web-proxy mode; the interface presented to clients is consistent.

In this chapter, eleven prototype experiments are discussed that encompass a number of different models. The first four experiments described in section

6.1 explore different patterns for decoupling the various components of the DLS. Sections 6.2.1 through 6.2.3 discuss the use of tuplespaces as shared, ephemeral link resolution results caches. Sections 6.2.4 through 6.2.6 consider the role of Directory Services as an appropriate distribution model, and section 6.2.7 discusses a prototype pDLS that utilises a virtual environment gaming model.

6.1 Decoupled pDLS model

One enactment of the proposed framework would be for each individual participant to run and interact with precisely one local pDLS instance that served as both query dispatcher and link resolver, c.f. the original DLS.

Where the user has selected remote linkbases as targets for link resolution queries using the GUI, their local pDLS proxy dispatches individual queries over the network to the remote user's (named) pDLS instance.

The resulting communication pattern between pDLS instances would appear as shown in Figure 6-1.

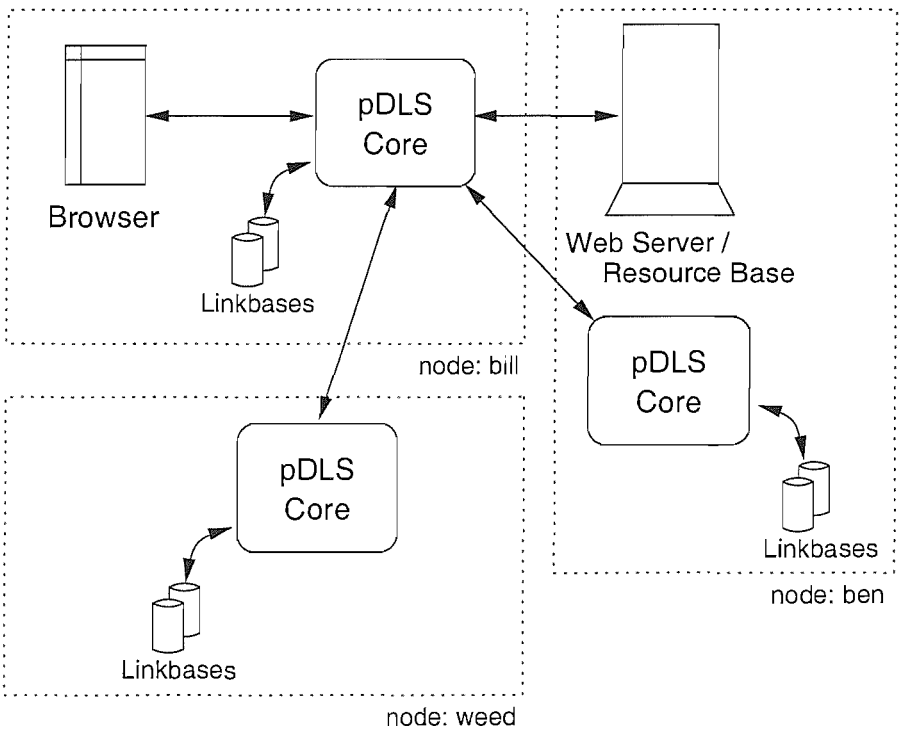

Figure 6-1: Decoupled single-process pDLS

This fits well within the proposed framework and is an entirely valid, if inefficient, approach to distributed link resolution within the target scenarios. There is one process for querying, resource transport, local resolution (match against local link database), remote resolution (CGI-like query of remote link resolution process on a per-term basis), and result aggregation and presentation.

An alternative approach would be to decouple the DLS into its constituent functional parts: the query interface, the transport proxy and the link resolver. Each client pDLS instance would then conceptually resemble Figure 6-2.

Once decoupled, each of the participant components considered together form a "distributed" DLS, where communication between these peer components is hidden away from the end users – of which there may be many (each of the physical participants in the application).

Popular file-sharing approaches to peer-to-peer content distribution systems are appropriate for consideration as pDLS implementations, however they have not explicitly been developed for within this research. In part this is a moot point in that all of the approaches discussed here are peer-to-peer in the sense that there is communication peers of equal role and stature (*vis a vis* the client-server relationships of Web servers and user agents).

The lack of explicit development with popular peer-to-peer file-sharing systems is due to several considerations, most notably that it is the communication pattern between components that is under consideration as opposed the implementation, and systems such as Jxta[8] and Gnutella[9] employ unicast and simulated-multicast (that is, unicast to a set of nodes) communications between nodes – both of which we consider here.

[8] Project JXTA: http://jxta.org/

[9] Gnutella: http://gnutella.com/

Also, and particularly in the case of Gnutella, the sole purpose of the peer-to-peer platforms is to shift content about a wide-area, relatively static network; whereas our focus is on local networks whose topology is flat, and connectivity ephemeral.

The subsections that follow document four experiments that examine different decoupling patterns and their relative merits within the suggested framework.

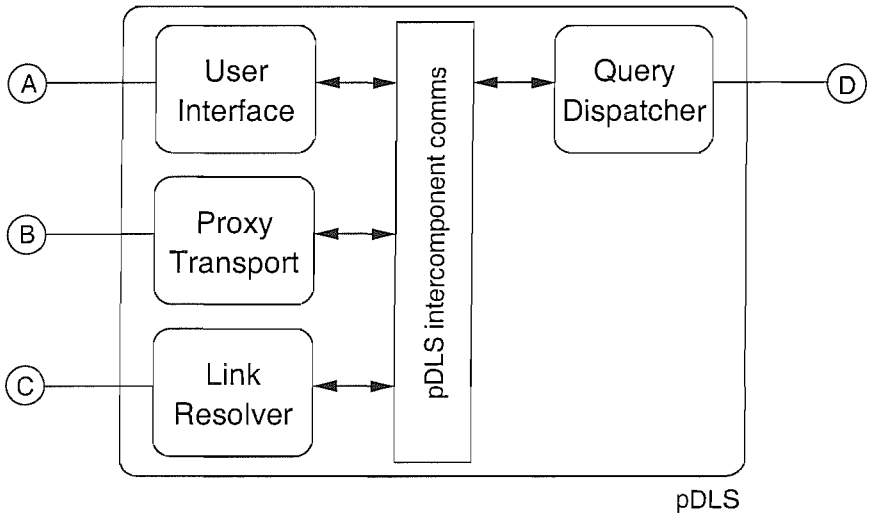

Figure 6-2: Component-based pDLS

6.1.1 Unicast IP

Unicast communication is where information is addressed explicitly for one particular recipient, for example, as with postal mail. A typical device or device configured for participation in an IP-based network would be configured with an address at which it can be assured to be the only recipient of data addressed explicitly to it.

In the context of the scenarios, zeroconf technologies provide for link-local IP addresses to be configured as nodes join the scenario in the case of no infrastructure-provisioned addressing mechanism (such as DHCP). In combination with the local area domain name resolution service, mDNS, this means that symbolic host names and local IP addresses will exist for all

participating nodes in the framework, and therefore unicast socket-based communication will be possible.

As discussed in section 2.5.11.2, early work has decoupled the monolithic DLS architecture into a distributed one where the Web proxy component is distributed from the link resolution engine. However, the communications and process architectures have not been discussed in the context of the scenarios of interest to this research.

Where the transport proxy component of the DLS communicates with a singule link service proxy, an alternative architecture would be for the proxy to communicate to a group of link service resolution components. Each component could then process the query data individually, returning their results for aggregation, filtration, and delivery to the user.

6.1.1.1 Architecture

The architecture of the pDLS link service distributes the different processes of link resolution into a query dispatch and result aggregation component that arbitrates queries of the various enabled linkbases.

The Link Resolvers in this prototype resemble fully-fledged DLSes in that their exposed interface offers a HTTP transport for linkbase query and update in an interaction resembling HTTP GET queries. Linkbase resolvers are identified as discussed in section 5.2.5, where the linkbase identifier URL also identifies the address of a serviceable resolver.

An example interaction is as follows.

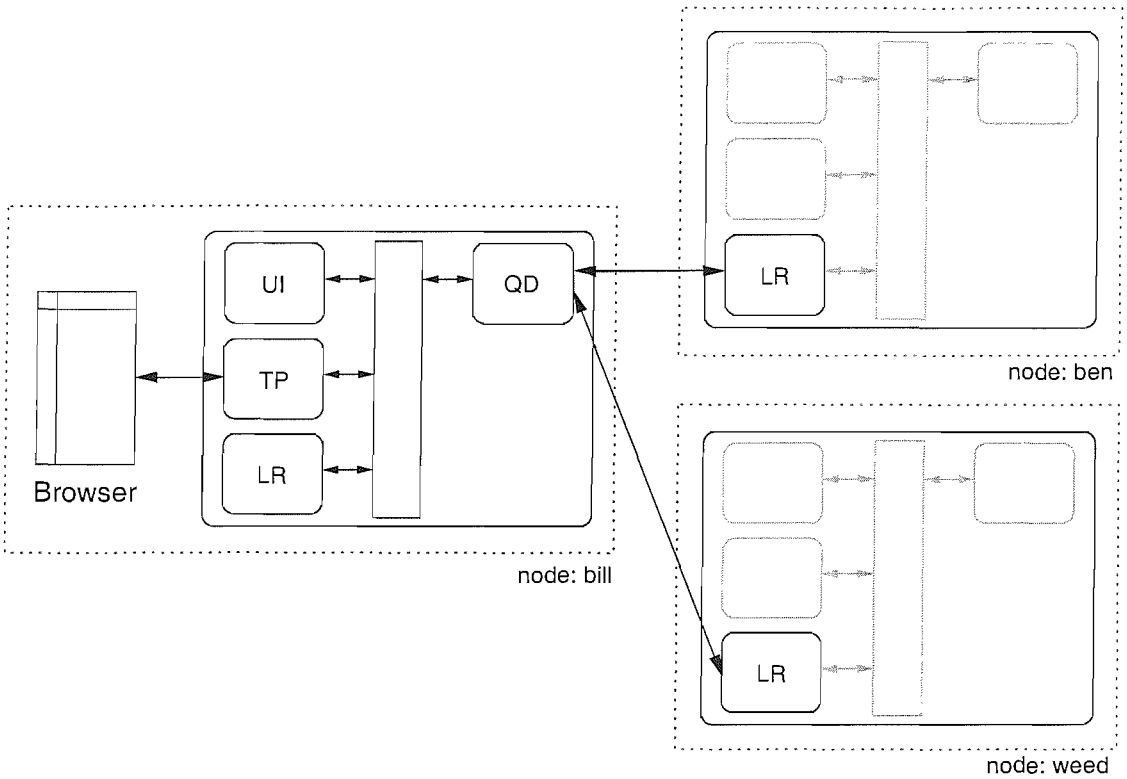

Figure 6-3: Distribution with Unicast IP

The pDLS client on node `bill.local` navigates to a resource in their Web browser by selecting a remote resource discovered using one of the techniques discussed earlier. The browser has been configured such that all HTTP traffic is routed through the local pDLS transport proxy, and therefore the proxy's Query Dispatcher sub-process is responsible for retrieving the resource from the remote resource base. According to the configured strategy (see section 5.2.2.1) the Query Dispatcher collects terms from the resource and distributes to link resolvers enabled in the GUI over iterative unicast connections.

Link Resolver processes return any successful resolutions back to the invoking Query Dispatcher, which then determines which links to present as Available Links in the GUI, with associated metadata (e.g. which link resolver returned the result).

At this point, the pDLS Query Dispatcher might choose to attempt link composition on, for example, non-local link destination anchors in an attempt

to discover if there are any local representations of the remote (and probably un-navigable) resource.

6.1.1.2 *Relative merits*

With regards the 'monolithic' single-component approach above, the following observations are made of the Unicast IP approach to pDLS distribution:

Pros

- Distribution of functionality
- Introduction of third party services
- Concurrency and Aggregation
- Increased scale

Cons

- Increased latency
- Increased network traffic
- Interaction of Utilities

Distribution of functionality

By farming out link resolution from the user-facing service to individual processes, different link service implementations can be swapped in or out without rewriting large pieces of the pDLS code.

Introduction of third party services

Developing the notion of pluggable-resolver functionality further leads to the observation that alternative services can be integrated into the pDLS, abstracted by a common query interface. For example, the Google Web search engine offers a similar interaction style to the pDLS (pass in query, receive set of links) and so could be treated as a link service component in that respect, wrapped as a service exhibiting the same interface as the other resolver processes in the system.

Concurrency and Aggregation

Distribution of the resolution process to individual processes working in parallel, perhaps on different nodes of the network, enables multiple linkbases to be queries concurrently, with a clean separation between them.

Implementations in which different linkbases are queried by the same resolution process typically do so sequentially, either by consulting each linkbase in turn for a given query, or by performing each possible query on each linkbase in turn.

The process of then aggregating the results from concurrent queries into a unified result set for delivery back to the client then befalls the process that distributed the queries. In the example of Figure 6-3, the Query Dispatcher (QD) on node `bill.local`.

Increased scale

Related benefit to concurrency in resolution processes, an architecture that permits additional resolution processes to be added to the system without impeding significantly on interaction time increases the scale of the system as a whole.

Increased latency

A monolithic link service features linkbases local to the resolver, and typically uses API calls to retrieve data, e.g. by loading the linkbase into memory and querying the data structures directly, or using a local database engine. The moment that query data needs to be distributed to other processes via network sockets, the latency introduced becomes more noticeable.

The 'remote' processes could be local to the machine, and therefore the latency minimised to intra-machine inter-process communications, comparable to the query of a local database engine (the creation of a packet, delivery to the local IP stack and then delivery to the receiving application from the same IP stack).

Alternatively, the processes could be on remote nodes and thus the additional latency inherent in delivering packets across the network is introduced. Keeping the remote processes as topologically local as possible to the source of the query would serve to minimise this latency, as is the case in a strictly link-local flat network as has been assumed within the scenarios.

Increased network traffic

For each resolver process that is on a separate node to the Query Dispatcher, all query data (and potentially all resource data depending on the term definition strategy employed by the pDLS) needs to be transported to the remote process. With, for example, four separate resolver processes and a query that was one kilobyte of data, at least four kilobytes of network traffic would be generated just to distribute the query to the individual resolver processes.

The inbound traffic is indeterminate, too. Resolvers with no link or linkbase results would generate a minimum of reply traffic, although in contrast, other resolvers with a large amount of result data may generate increasingly large amounts of traffic.

Interaction of Utilities

With this approach to distribution, the interaction with utility processes that require a degree of awareness of the link service's actions (e.g. a query or results cache) need to be explicitly informed as an additional process of the participating components.

A 'hooks' approach to the Query Dispatcher and Link Resolver processes would enable utilities to register to become part of an informed chain of processes at the expense of increased latency and load on the subscribed processes.

6.1.2 IP Multicast

IP Multicast communication is a technology that allows a source node to send data to multiple recipients without addressing each of them individually. Source nodes only need to send their data once, and in multi-link networks, multicast-capable routers distribute the data forward to other networks that have explicitly expressed an interest in the material. IP Multicast was proposed by Deering (Deering, 1990) as a mechanism that would reduce the sheer number and therefore bandwidth requirements of many unicast point-to-point connections distributing, for example, videoconference data between sets of nodes.

Not only does the use of multicast dramatically decrease load over transit networks between communicating sites, but it also provides an excellent abstraction mechanism for addressing nodes. There is provision in the various IP network programming interfaces for nodes to join multicast 'groups', specifically reserved addresses within the IP address hierarchy for use in multicast communication. The addresses are treated such that any traffic transmitted to a particular address and port number pair is propagated to participant nodes, receiving packets using the same socket-programming interface as they would for typical Internet communication.

Where data transport protocols such as HTTP are reliable, with packets guaranteed to be received in the order which they are sent and without loss through the use of TCP (Transport Control Protocol), Multicast datagrams are sent as Unreliable Datagram Protocol (UDP) packets, and so are susceptible to loss due to network conditions. It is the responsibility of the higher layer (read: application or middleware) protocols to either cope with or recover from the potential loss or jitter-delayed nature of the data. However, within a local network so long as the size of the datagrams fall within the link maximum transfer unit, the reliability of UDP should be reasonable for small, single-shot data, such as link resolution queries.

Where Internet-scale use of multicast typically features audio or video-conferencing and applications that offer collaborative functionality such as shared whiteboards, local- and enterprise-scoped IP Multicast has seen good

use for other, less interactive applications, for example, software distribution within an enterprise.

In the context of the identified scenarios for link services, it is the ability to deliver datagrams to collections of nodes without explicitly addressing any individual one that offers interesting possibilities. Considering the use of IP Multicast to distribute query data to resolver nodes, the following architecture results.

6.1.2.1 Architecture

The method of operation for the prototype developed to exercise this approach is as follows:

The client-facing interface and functionality of the Query Dispatcher remains unchanged from the decoupled-unicast approach above.

All components (Query Dispatcher and Link Resolution process nodes) join a well-known multicast group. For the purposes of experimentation, the experimental group 224.0.0.252/5465 was chosen due to its current unreserved state within the Internet Assigned Numbers Authority registry for link-local multicast groups (Albanna, 2001; IANA, 2001). The dispatcher distributes query data to the multicast group such that link resolution processes receive it, process it and then return their results.

This approach requires an additional change in the Query Dispatcher process in that the underlying network transport has changed from HTTP-over-TCP, a streamed, delivery-assured (from the perspective of the application) transport to UDP. UDP is a connection-less transport, and is classified as unreliable in that packets can arrive in a different order to which they were sent, or even lost, due to existing conditions within the node's network stack or other traffic between originator and receivers.

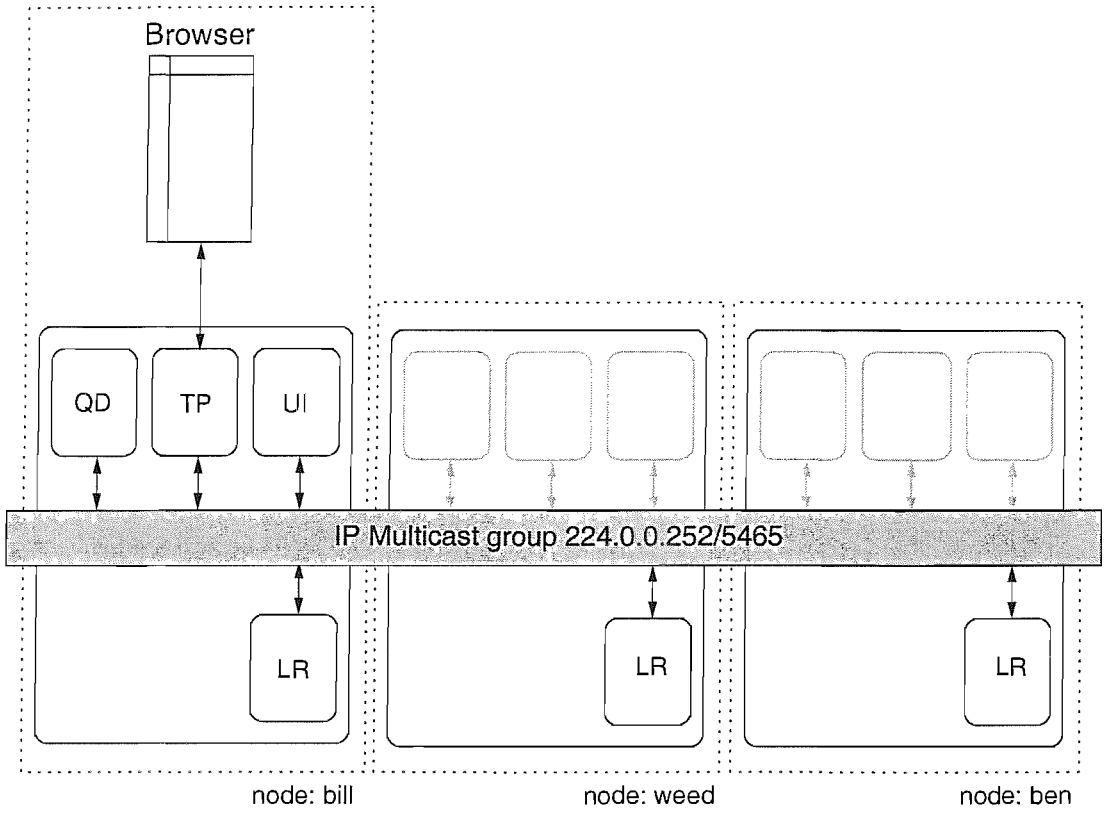

Figure 6-4: Distribution using IP Multicast

Through assuming a single link network (i.e. no routers present), many of the causes of UDP unreliability are removed. However, there should not be any reliance put on the network protocol stack local to the Query Dispatcher or of the destination Link Resolution nodes that datagrams will arrive in the order that they were sent, or that they will arrive at all. This could, for example, be due to sheer network load where UDP packets are discarded in preference to streamed, TCP packets (e.g. due to Quality of Service and queuing preferences within the stacks). The goal, therefore, should be to get the entire query into a single datagram so that the Query Dispatcher can truly 'fire and forget'.

The requirements resulting from the scenario analysis identified 802.11[ab] wireless networks and fixed wired Ethernets as the target environment for the pDLS experiments. The maximum IP packet size over these links is restricted by the logical link control layer, which is the standardised IEEE802.2 protocol in both cases. The maximum transfer unit (MTU) for the entire frame (i.e. the IP packet's header, the transport layer's header and the payload data) is 1492

bytes. Other local area networking technologies that might be seen in similar scenario deployments include Token Ring, which has a larger frame MTU of 2 kilobytes, and therefore choosing a maximum query size in the Query Dispatcher that is smaller is not an issue.

A pDLS query transport therefore needs to fit the link resolution request within this limit. In existing implementations, the query transport is HTTP GET requests, with linkbase selection, partial link data to match on and additional HTTP protocol headers all combining to compose a query. Where the HTTP protocol specification (RFC2616) limits the path and query components of a URL to 2048 bytes, our limit has to be significantly less for encoding link query data (approximately 1470 bytes, allowing for transport headers).

The underlying IP protocol is datagram-based and also unreliable (a 'best efforts' service) and the TCP transport provides assured in-order reliability atop of that unreliable medium. It is possible to implement TCP-like delivery assurance controls over Multicast UDP within the Query Dispatcher at the expense of the additional code, state and computation at the Query Dispatcher nodes.

A difference between this approach and the Unicast approach above is that there is facility within the communication pattern for the result set to be distributed in different ways, depending on a resolver strategy: The result set could be unicast back by each link resolver process to the query dispatcher using a TCP stream socket, which would benefit from being a reliable, order-assured connection with assured delivery so long as the node is present on the network.

Alternatively, the resolvers could use IP Multicast to distribute any successful link resolutions such that other processes (e.g. other resolvers, cache utilities, third party event viewers) could be made aware of the results, with due consideration to the transport reliability constraints expressed above.

6.1.2.2 *Relative Merits*

The relative advantages and disadvantages compared to the unicast approach above are as follows:

Pros

- Simpler server (distributor) code
- Network efficient
- Engagement of Utilities

Cons

- Privacy
- Potentially Lossy
- Partitioning

No relative change

- Latency

Simpler code requiring less state and awareness

The use of IP Multicast means that processes that distribute data to many nodes do not need to maintain a list of the nodes to which they are sending data, nor is it necessary to open and maintain sets of connections to each individual processes. This makes the task of the query dispatcher significantly simpler to code, and facilitates extension to a greater number of remote resolver processes without further modification to the dispatcher.

Network efficient

With individual, unicast socket connections to distributed resolver processes, n processes required $n.b$ network bandwidth to distribute the query. Subsequently, $n.b$ connections will result for the response streams as resolver processes report their results. Using IP Multicast to distribute the query to the resolver processes means that only one copy of the data traverses the network.

The choice of whether resolver processes unicast their results back to the aggregating proxy or whether to multicast them back can be informed by the desired functionality of the individual resolver processes. If it were the case that the result of link resolution events were desired to be cached (or at least observed) by other processes in the system, then the use of IP Multicast would readily enable this without any awareness by the processes generating the results; nodes that wanted to listen could join the multicast group for the result set and act on what data they deemed appropriate. This has implications on the privacy issue discussed below, but makes more efficient use of the (possibly scarce, in the case of lower-bandwidth wireless) network resource.

Engagement of Utilities

The relative anonymity afforded and lightweight subscription mechanism of multicast group membership means that additional processes, such as the Utilities, can subscribe and participate in the various process interactions with ease.

Privacy

When using unicast connections between components, the data stream is relatively safe from casual prying eyes. Switched networks stop casual snooping of unencrypted network traffic by virtue of the active network equipment only forwarding frames that are either explicitly unicast to individual node addresses (at the Ethernet layer), or are marked as for network broadcast.

A side effect of the simplicity of joining a multicast group means that it becomes trivial for other malicious or otherwise unwanted processes to be able to receive data sent to the group of processes, for example resources being browsed, link resolution events being reported, etc.

Techniques for provisioning privacy and authenticated access to data transports such as SSL (Secure Sockets Layer) (Dierks, 1999) do not apply to

UDP datagrams for they are connectionless. Whilst it is feasible to encrypt each individual datagram before publishing it to a multicast group, the additional computational and coding overheads make such a tactic prohibitively expensive and not worthwhile.

A candidate technology might be to employ a symmetric-key cipher that is relatively fast, computationally far less expensive than most alternatives, and requires a 'shared-secret' key to be held by all participants. The issue then, aside from increased latency and overheads, would be how to distribute the key to all the (desired) participant nodes.

Potentially Lossy

Whilst unlikely in local area networks where datagrams are kept smaller than the underlying link's defined maximum transfer unit, the use of IP Multicast requires the use of connection-less communications between nodes. UDP packets are susceptible to packet loss, and, more importantly, packets not guaranteed to arrive in the same order in which they are sent.

Any stateful monitoring or delivery assurance would need to be coded explicitly, whereas with TCP these guarantees come for free.

Partitioning

The adoption of a single multicast group for all component communication results in any requirement to partition the information space becomes something that the component has to explicitly manage. For example, a process might only wish to receive link resolutions from a particular resolver instead of hearing all communications from every component in the system.

Different multicast groups for different purposes or sub-partitions of the space would provide a simple mechanism for this partitioning, and decrease the amount of processing time wasted by the determination of the relevance of incoming messages from the network at every component. However, the

issue then becomes how to advertise those additional groups away from the 'Well Known Service' address chosen above.

Latency

Compared to the unicast-decoupled approach, latency is no different as a result of the move to an IP Multicast architecture, except that the use of a connectionless transport for query dispatch removes the connection setup times of the individual TCP socket connections between components.

6.1.3 Application level Multicast

The notion of multicast described above is that which is provisioned at the IP layer of the network stack, as a service of the underlying networks. An alternative approach is to 'simulate' network-layer multicast within applications, e.g., by maintaining lists of interested nodes and explicitly unicasting information as it becomes available, e.g. by some subscription metric.

To do this on a per-application basis would be expensive and require significant design effort to ensure that issues of discovery, membership, interaction, delivery and scale (amongst others) were all catered for. Rather, a middleware network service should be employed that meets the requirements without great additional overhead.

There are a number of different systems that provide coordinated multicast-like delivery of information to sets of participants, of which two representative approaches are discussed here and prototyped for link services.

6.1.3.1 Publication/Subscription using Channels or Topics (IRC)

The IRC (Internet Relay Chat) protocol has been designed over a number of years for use with text based conferencing across the Internet, standardised by the Internet Engineering Task Force in 1993 (Oikarinen, 1993), and a

predecessor to modern Instant Messaging systems such as Yahoo IM, MSN Messenger and ICQ.

IRC employs a number of different message distribution models, the most commonly observed of which is a publish/subscribe many-to-many model in which clients join 'channels' that serve to cluster different themed conversations, or 'topics'. Servers host channels, providing the necessary message multiplexing and management functions by keeping track of the channel members and their location.

The IRC Protocol itself is based on the client-server model, where a typical setup involves a single process (the server) forming a central point for clients (or other servers) to connect to. The primary role of the server is to perform the message delivery and multiplexing functions.

Many servers can be interconnected, providing a wider-area distribution for messages. Clients connect to local servers, that function as multiplexing nodes so as to reduce the number of simultaneous, low bandwidth but interactive sessions across the Internet that would otherwise result from an architecture with a single server.

IRC servers provide points to which clients may connect to so that they can talk to each other, and also for other servers to connect to. The server is also responsible for providing the basic services defined by the IRC protocol. The only network configuration allowed for IRC servers is that of a spanning tree where each server acts as a central node for the rest of the network it sees. The simplest and smallest IRC network is one that consists of only one server.

There are two types of clients of IRC networks, User Clients and Service Clients.

- *User Clients*, 'Users'. Programs that provide a text-based interface used to communicate, 'chat', interactively
- *Service Clients*, 'Bots'. Service clients are software processes that provide ancillary functionality to the IRC network, for example the

maintenance of administration or accounting information (e.g. logging where clients have connected from), or the provision of extra services beyond those specified in the IRC protocol, such as file sharing ('fservs'), conversation archiving, AI research exploring the Turing test for artificial intelligences (e.g. based on interaction with Eliza (Weizenbaum, 1966)) and entertainment, for example 'loki', an IRC-based Texas Hold'Em poker dealer (Papp, 1998).

The only interaction permissible for clients is with their immediate peer server, which then relays communications on to remote clients or servers as necessary. By virtue of enforcing a spanning tree topology between servers, the protocol ensures that there is exactly one route from one client to any other participant client in the network, and that route is the shortest path between any two servers in the tree. Whilst this enables a simple routing mechanism for servers, it does not promote tolerance to individual link failures between servers; should such a fail occur, a 'net split' results and the IRC network becomes partitioned into separate networks. In this case, clients on either side of the failed link can talk to each other on the same side of the link, in the same channels etc., but not to former peers that still exist in the same channels on the other side of the link.

Clients of the IRC network have five communication patterns that they can employ, of which the first two are most commonly seen in Internet deployments:

- **Private**. There is provision for 'private' messaging between two individual clients. Messages still route through the server topology, even if both clients are connected to the same IRC server, however.

- **Channel**. The typical multicast pattern observed when clients post messages to a channel, which is then distributed to all other clients throughout the network that are also on that channel

- **Mask**. Similar to Channel, except the message is delivered to all clients whose host or server information matches a given pattern. This

provides a means to contact, for example, all clients currently bound to a particular IRC server in the network, in anticipation of it being disconnected from the tree.

- **List**. The most resource intensive and inefficient distribution pattern, List enables clients to explicitly name a set of remote clients, channels or masks that a particular message should be distributed to.

- **All**. Broadcast message to all connected clients, irrespective of channel or server bound to. In modern deployments, a form of this pattern that would broadcast to all 'operators' (clients with administration privileges) was sufficiently abused that the feature is usually disabled.

Architecture

Each bot process acting as communication conduit for the different pDLS components connects to the IRC server having first discovered its presence using a third party discovery tool. In the example system below, there are bot processes for the Query Dispatcher and each of the Link Resolver components, including remote ones that were initiated by the remote user on their arrival to the scenario.

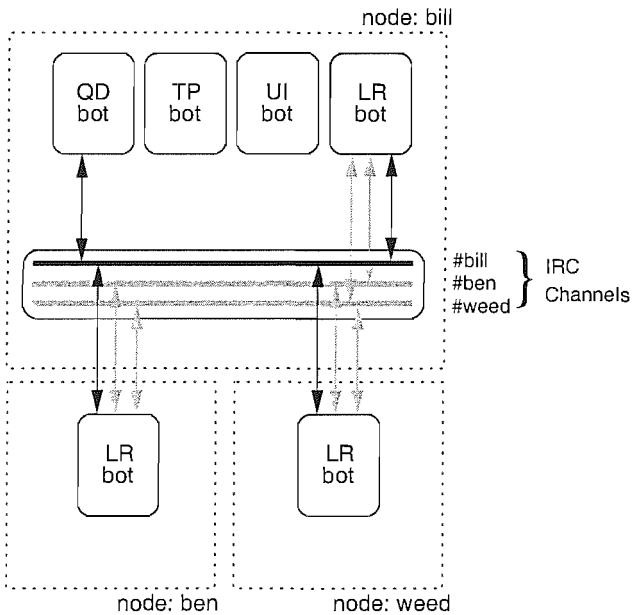

Figure 6-5: Distribution using Internet Relay Chat

Where the IRC protocol restricts the identifying token (the 'nick') of the client processes to an eight character ASCII string, the pDLS components employ a naming strategy that ensures (bounded) uniqueness by selecting a substring of their node's host name that is guaranteed to be unique due to underlying zeroconf networking restrictions, an instance serial number for cases where more than one node on the network has the same substring, a function token (e.g. 'r' for link resolver process), and a function token serial number such that more than one instance of the particular process can execute on each node.

This crude but effective strategy for naming components within the context of the messaging middleware does have a relatively high collision rate in comparison to node name-port number that would otherwise be used to identify instances of components in the IP Unicast and IP Multicast experiments above. However, it is deemed sufficient for the sizes of networks and numbers of components anticipated within the scenarios and serves to enable the exploration of IRC as a group-based messaging middleware for the pDLS framework.

Having connected to the IRC server and selected an identifying name ('nick'), the bots then join, or create as a process of joining, IRC channels that correspond to their node's name, e.g. the channel `#weed` for the pDLS interface for the user on node `weed.local`. Bots may also other channels for debugging and command and control purposes within the framework. Remote components, i.e. those operated by other users, can be included in the query operations by using IRC's 'invite' mechanism for clients to be drawn into a channel so as to be able to see and contribute to communication on it.

This particular architecture experiment considers the use of channels as a mechanism for individual users to collect service interfaces, a user-centric 'view' on the framework. Other arrangements within the middleware are possible, for example where each resolver component has its own channel, and query dispatchers join the channel for each desired resolver. The resulting architecture has similar communications characteristics as the one trialled here.

Query Dispatcher processes publish link resolution requests to the channel, and all of the other members can react by resolving the query and publishing any resulting successful link resolutions back to the channel for the Query Dispatcher to collate.

Additional housekeeping and other pDLS infrastructure messages can be transported in a similar way without interfering or impeding the link resolution process. This is achieved through the definition of an on-channel messaging protocol, comparable to the use of path specifiers as parameters passed to the Link Resolver's HTTP-like query interface in previous experiments.

The query to resolve a specific term ('clipping') and source anchor ('report.pdf') in the previous experiments would appear as an HTTP request resembling that shown in Figure 6-6.

```
GET /r?id=0xea&t=clipping&sr=http://bill.local:6970/r/report.pdf
```

Figure 6-6: Example HTTP Link Resolution Request

The same query appearing as a message on-channel with this IRC-based middleware would appear as the XML message:

```
<resolve-request id='0xea'><src><term>clipping</term><resource>http:
//bill.local:6970/rb/hockey/report.pdf</resource></src></resolve-
request>
```

Figure 6-7: Example IRC Link Resolution Request

Likewise, where a valid link resolved response in the previous HTTP-based experiments would be:

```
<resolved src="http://ben.local:6970/lb/iihfrules">
<link id="CLIP">
<src>
<resource/>
```

```
<offset/>
<sel>clipping</sel>
<sel>clip</sel>
</src>
<dest>
<resource>http://www.refline.net/7162/72448.html</resource>
<offset/>
<sel/>
</dest>
<owner>http://ben.local/concepts#rusty</owner>
<time-stamp>2001-07-17T15:01:47+0100</time-stamp>
<title>Player Penalties: Clipping (CLIP)</title>
</link>
</resolved>
```

Figure 6-8: Example HTTP Link Resolution Response

With the IRC approach, the entire resolved link result as in the figure above would be encapsulated with an identifying element, thus:

`<resolved-link id='0xea'>` *(contents of Figure 6-8)* `</resolved-link>`

Figure 6-9: Example IRC Link Resolution Response

The use of XML as a message encoding has arisen in part to the readily available tools to parse message contents, to enforce structure and content so as to reinforce behaviour, and to enable through the use of XML message schema a level of shared understanding as to the nature of the messages and their content.

Relative Merits

Considering an architectural approach using a publication/subscribe channel-based mechanism such as IRC against the previous prototype architectures suggests the following:

Pros

- Extensibility
- Domain Partitioning
- Debugging/Monitoring Capacity
- Multi-party
- Security (within-model)

Cons

- Transport Characteristics
- Service Discovery
- Service Provision
- Fixed Network Model
- Security (privacy)

Extensibility

Adding additional components to the system is trivial. The set of client operations is well defined, and the channel model for clustering communication between components with similar interests is very flexible.

By being designed for 'free speech' between human participants, there are no restrictions on the content of the messages communicated, other than addressed in the Transport Characteristics section below. The responsibility of defining a protocol for messages conveyed using IRC that is understood by all components is left with the system developer.

Domain Partitioning

The channel facility enables the partitioning of the communication space in a similar manner to the different IP Multicast groups, as discussed above.

For example, it is possible to create separate channels to represent different linkbase themes (see section 2.5.11.3), and then have different link resolver components join the various channels to which they have relevant links available.

The issue then becomes how to convey the intended content of the various channels so that processes or users explicitly publishing content know which ones to join. Within the channel architecture of IRC, there is provision for a free-form channel description. This can be used to convey metadata regarding the channel, for example, an understood theme or concept label, which then would need to be reflected in the pDLS GUI.

Whilst the ability to partition domains is advantageous, it does introduce an additional complexity in the Query Dispatcher. The process needs to be aware of what themes are available, in order to join those channels as a client and so publish to and collect data from.

Debugging/Monitoring Capacity

Using IRC as a communications substrate means that debugging or in the general case monitoring the interactions between the various DLS components is trivial. In order to observe the various interactions, one only has to join the server using an off-the-shelf IRC client and join the pertinent channels.

This also provides an interface through which, as a system developer, interactions can be fabricated and so functionality of individual components tested without requiring significant supporting infrastructure. Knowing the nature of the messaging protocol used between processes over IRC provides the ability to 'pretend' to be a query dispatcher, or any other component of the system and, for example, replay captured interactions.

Multi-party

One of the issues with the IP Multicast implementation was that it was difficult to track the source of a query, for the communications were keyed solely on the IP address of the sender, particularly when there was more than one participant process on the same node (therefore with the same IP address).

With IRC, each connection client has the notion of a unique identifier (a 'nick' in IRC terminology), and therefore the ability to track the origin of communication is straightforward.

Security (within-model)

The IP multicast distribution model did not provide any notion of secure access to data without the adoption of an application-layer encryption technology.

IRC offers simple but effective channel-level access authorisation by the use of 'channel keys'. In channels where a key is set, only those clients that specify the correct channel key at connection-time are permitted to join the channel and therefore receive data intended for that channel. Whilst this suffers from the same problem as any shared-secret approach to security, namely that of key distribution, it does permit a simple level of access control.

Transport Characteristics

The IRC protocol permits almost all 8-bit bytes as transport data, however some of the ASCII set are reserved for protocol use, particularly the null byte (ASCII-0), carriage return and line feeds (ASCII-13 and -10 respectively). Whilst this is fine for transporting text-based queries and link results, any query data that requires binary transport will require the use of a out-of-band transport, encoding before dispatch, or the use of a different transport. Binary query data would include any non-textual media for matching, for example image regions and textures, music, and video.

Out-of-band transport would be inefficient, as the dispatcher would have to be aware of each of the resolver processes to which the query data needs to be sent, which is contra to the point of using a publish/subscribe, decoupled messaging middleware.

Pre-encoding by a technique such as the Unix tool 'uuencode' would enable binary query data to be distributed to the various resolution processes. Not only does the encoding process result in the data being exploded by virtue of the encoding process packing data to a smaller character set and therefore requiring additional data, which decreases the efficiency of network usage, but the data will need to be decoded at each individual receiving process, a significant and unnecessary computational overhead incurred by many processes.

Service Discovery

IRC servers are typically deployed in stable networks, their addresses well advertised and the links between different servers within an IRC network static.

There is no standardised notion of service discovery for an IRC server, beyond the casual approach of using the '`irc`' domain label as a service alias. For example, the node named `irc.webcentre.net` offers an IRC service on the well-known port (as defined by the Internet Assigned Numbers Authority) 6667/tcp, c.f. the use of domain labels with the World Wide Web.

The approaches discussed in section 4.5.4 can be applied to provision service discovery within the context of the scenarios, however the process of defining and maintaining inter-server links is more troublesome due to the fixed network model, discussed below.

Service Provision

The nature of the IRC system is that there has to be at least one IRC server process present throughout the life of the scenario, and therefore this is well suited to environments where some level of permanent infrastructure provision is available.

Where this is not possible, for example, in truly infrastructure-less environments such as alien meeting rooms and conference venues, then this vital infrastructure component has to be operated on at least one participant node.

With consideration to the fixed model (discussed below), this can be achieved in part by the first participant to arrive at a space attempting to discover a live IRC server, and starting one locally if none appears available. However, should that client then wish to leave the scenario before all of the subsequent participants, there is no clean mechanism to 'hand over' the IRC server or its connections to other server instances.

Fixed network model

Another issue with spanning tree model is that it is difficult to add additional servers to the network without significant overhead, including the reconfiguration of other servers.

It is possible to add servers to a live IRC network on the fly, reconfigure the necessary connection parameters in the peer servers, and still maintain live traffic. However, the spanning tree configuration is handcrafted, and during each affected server refresh and subsequent global state learning process, there is a great risk of clients being disconnected, or not all clients receiving all messages transferred during the period of instability.

Security (Privacy)

There is no session encryption option for traffic between IRC clients and servers, which results in all traffic being susceptible to snooping and tampering.

It is possible to encapsulate the transport stream between nodes in an IRC network in Secure Socket Layer (SSL) connections. Whilst developed as a technique for securing the HTTP transport for Web traffic, the approach is generic enough that any stream-based protocol can be 'tunnelled' over insecure media.

For the purposes of securing IRC client connections, SSL does offer a mechanism for countering transport privacy and authenticated access to the transport data. The overheads incurred in authenticating components are one-off, only at connection time. The run-time encryption overhead, however, would be significant for smaller devices where processing resource is scarcer.

6.1.3.2 Content-based messaging (Elvin)

IP Multicast and IRC offer models of group shared-medium communication, where all participants subscribed to a channel or group see all messages published. The model employed by middleware architectures such as Elvin

(Segall, 1997; Fitzpatrick, 1999) is to route each individual message dependent on its message *content*.

Whilst similar in conceptual architecture to other publication-subscription messaging systems such as IBM's MQ Everyplace (IBM-MQe, 2001), a fundamental difference of is in the relationship between the mechanisms by which consumers of notifications declare their interests. Elvin facilitates a flexible pattern-matching model that can inspect message content, compared to the topic-based approach of systems such as MQe.

A suitable conceptual example is to consider how the message router inspects the message to be delivered: systems such as MQe can inspect the 'envelope' that specifies a queue to post to, whereas systems like Elvin interrogate the 'body'.

Clients register patterns that describe the attributes of messages that they wish to be informed of, codified in a subscription placed upon a message router (Segall, 1997). Processes that report on events register the fact that they are *message producers* with these routers and, when an event is to be reported, invoke an 'inform' performative on the router, which then forwards the message to all clients whose registered subscriptions match.

Similar to IRC, Elvin offers the facility to create an overlay network of federated message routers, providing capability to distribute the 'reach' and message scalability of the system. Through only forwarding notifications toward peer dispatchers themselves with clients interested in the notification's content, the amount of redundant communication is reduced.

Elvin has been designed with a lightweight, local-area network router discovery process built-in. As long as the network on which the service is running is broadcast-enabled – which the target scenarios have been assumed to be – clients can easily discover routers to interact with. This contrasts with the IRC implementations that required pre-existing infrastructure to be configured, and specialised mechanisms introduced for discovery.

Notifications with Elvin are anonymous in that there is no service-provisioned mechanism for determining the source of a notification, nor is there a way to explicitly address a subset of bound clients. Should the clients all have the same subscription pattern, they will all receive the same notification, hence *content-based notification*.

Naturally, this feature can be deliberately misused by adding explicit data deliberately to the notification tuples to realise a unicast-like addressing mechanism. The coexistence of such 'faked' unicast communication with anonymous content-based notification within the same system provides for great flexibility in messaging patterns.

Other notification services exist with similar characteristics, including Keryx (Brandt, 1997) and Gryphon (Banavar, 1999). Elvin is chosen here as a representative candidate notification service due to its numerous and lightweight client bindings, an expressive subscription language, scalability through router federation, and integrated router discovery protocol. Analyses of other notification frameworks can be found in (Carzaniga, 2001) and (Rosenblum, 1997).

Architecture

The architecture of the Elvin prototype is similar to the IRC approach above, each client component of the messaging substrate maintains a TCP socket connection to an Elvin message router but the client interaction with this router is significantly different.

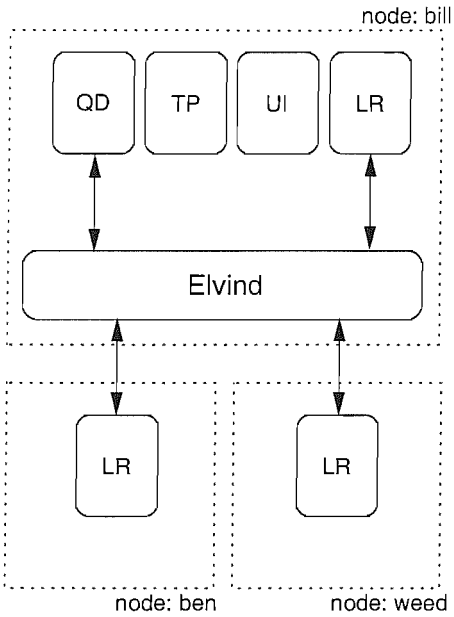

Figure 6-10: Distribution using Elvin

Each process binds to the message router, which has been located using Elvin's provided discovery mechanism. Subscriptions for events of interest are registered: concerning the primary pDLS hyperstructure components, the Query Dispatcher is primarily concerned with link resolution events, and the Link Resolvers are to receive notifications that contain link queries.

The Query Dispatcher responds to user navigation by informing (publishing) query notifications as a result of extracting terms to anchor on from the requested resource.

Link Resolver processes whose subscriptions match the asserted notifications, for example on linkbase name or a particular type of resolution query, receive copies of the notification and then enact the query contained against their linkbases. Any links resulting are then asserted as notifications of the appropriate format such that the Query Dispatcher (possibly amongst other processes) can collate results for presentation to the user.

Notifications within the Elvin system are bags of typed attribute-value tuples. One model used when experimenting with Elvin as a pDLS communications substrate was to have Query Dispatchers assert notifications with the following member tuples:

- `Transaction-Identifer`: *string* – A unique (within scenario) string comprising identifier of originating Query Dispatcher process
- `Query-Term`: *string* – The term being matched on, e.g. straight match against selection element of static links
- `Query-Source-Anchor`: *string* – The URL resource being inspected. As with all pDLS architectures, it is the job of the Query Dispatcher to select a strategy for which source anchor to query on, whether it is the URL of the resource in its current context (e.g. `http://bill.local/foo.html`) or in its global context (e.g. `http://example.com/blah/foo.html`) as coordinated by the available resource metadata (see section 5.2.7.2)

The various linkbases configured as enabled by the user determine which subscriptions are registered by the present Link Resolver processes. The mechanism developed for this within the framework has been for the Link Resolver subscriptions to be keyed on the Query Dispatcher (and therefore User) identifier specified in the Transaction-Identifier tuple. For example, the subscription pattern that matches all link resolution queries originating from the Query Dispatcher process on node `bill.local` would appear as:

```
begins-with('Transaction-Identifier','http://bill.local:6964/qd')
&& (require('Query-Term') || require('Query-Source-Anchor'))
```

Figure 6-11: Example Elvin Link Resolver subscription

When the Query Dispatcher on node `bill.local` publishes a query, the processes with a similar subscription to that above will receive copies of it. Any successful Link Resolution events are then published as notifications with the following member tuples:

- Transaction-Identifier: *string* – Same identifier as used in query, so that Query Dispatcher can mate replies with requests
- Resolved-Link-Term: *string* – Term that actually matched or was resolved against linkbase (usually the same as that queried with,

although some degree of term-processing may be employed by individual resolver processes)

- Resolved-Link-Destination-Anchor: *string* – URL of link target anchor. May refer to global resource rather than one available locally, in which case the Query Dispatcher may then choose to query local resource metadata catalogues for local representations
- Resolver-Identifier: *string* - A unique (within scenario) string comprising identifier of responding Link Resolver process, so that the Query Dispatcher can present the source of the link to the user when displaying available links

Query Dispatcher processes subscribe for notifications that have Transaction-Identifier tuples that match their identity, and then react to link resolution notifications as and when they arrive, matching results to queries as appropriate. An example Query Dispatcher subscription would be:

```
begins-with('Transaction-Identifier', \
'http://bill.local:6964/qd#') && \
require('Resolver-Identifier')
```

Figure 6-12: Example Elvin Query Dispatcher subscription

Relative Merits

The relative merits of the Elvin-based messaging substrate as developed for the pDLS link service are:

Pros

- Decoupled addressing
- Simple Client Interface
- Discovery
- Security
- Debug/Monitor

Cons

- Persistence
- Source Identification

- Service Provision

Decoupled Addressing

The fact that there is no explicit addressing mechanism with content-based messaging services such as Elvin means that participants need not maintain any model of group membership, or the overheads of explicitly having to identify targets for communications.

Having agreed, shared schema defining the notification tuple set means that participants can be sure that asserting notifications with particular attributes will be routed to other parties that are intended for receipt, and that those recipients will be able to understand the intended meaning of the tuple values.

Simple Client Interface

Participation in the messaging substrate is straightforward. Once a connection to a message router has been established, the only interaction for producers of notifications is an inform performative. Likewise, the consumer interface simply comprises a mechanism for registering subscriptions, and then through a callback mechanism, notifications that match the registered subscriptions are delivered.

Discovery

Elvin's in-built, IP multicast-based, discovery mechanism readily facilitates the location of live message routers for participants joining the network. The provision of scoped messaging domains, where different Elvin message routers can exist on the same network to provision different sets of clients, means that the provision of an Elvin message router explicitly for pDLS activity need not interfere with other uses of the messaging system already in use within the network.

Security

Elvin has facility for notifications to be encrypted using a symmetric cipher in all cases where data is transported across the network. Served as part of the client binding, this facility is transparent to client applications, and serves to enable privacy of communications between components. No level of per-participant authentication is provided, however.

Debug/Monitoring

Any process that understands the notification schema can register subscriptions such that they can receive copies of notifications as they are generated by the various pDLS components. This facilitates a debugging and monitoring interface, as it does the participation of other utility components within the framework.

Persistence

As with the approaches above, the use of Elvin as a messaging substrate does not offer any means of making notifications persistent, which means that processes have to be able to receive and process messages in a somewhat synchronous manner.

Alternative pub/sub-pattern systems such as MQ offer facility for 'reliable' notifications that are held in the message router such that processes that were unavailable or unable to receive the notifications at time of assertion and be made aware as and when they are ready.

Source Idenfitication

Notifications within the Elvin system are anonymous and therefore any operation that needs to know the source of a message requires an application-layer solution. In this case, this has been achieved by enforcement through the use of a schema that includes an identifying element in notifications. Subscriptions that match on that identifying element provision explicit unicast-like delivery.

Service Provision

Whilst Elvin provides a facility for a federated network of message routers, there is currently no mechanism for this internetworking to be updated dynamically. Likewise, Elvin provisions a level of service resilience by offering hot failover routers that can 'step in' and take over all live client bindings and subscriptions should the master router fail (i.e. leave, crash, lose network connectivity).

As with federation, failover cannot be configured on the fly, which means that the message router topology from an Elvin perspective would need to be known before subsequent nodes joined the scenario. This means that, as with the IRC approach to application layer multicast, either there has to be an infrastructure-provisioned Elvin message router, or that participants in the scenario have to maintain a live service between them.

Work is on-going providing an application-layer hand-over mechanism for participants such that, should a node that is currently serving as the host for the message router process wish to depart the scenario, another node can take over responsibility and all clients re-bind as appropriate. This should not detract, though, from the suitability of the content-based messaging approach for pDLS infrastructures.

6.1.4 Summary

Observing that it is the models of communication that are under examination and not the particular implementations, the issues regarding discovery and provision can be extracted and catered for externally. The appropriateness of the communication patterns as regards the pDLS framework suggest that the content-based messaging pattern has most benefit.

Decoupling the link resolution process from other functions of the link service such as term extraction, query dispatch and result compilation and presentation facilitates additional flexibility in the architecture as regards the integration of the other utility processes identified in the previous chapter,

and a clean interface for the inclusion of remote link resolvers present on other participant nodes in the system.

Scaling up from one Query Dispatcher (i.e. more than one user) has been possible with all chosen communication models, where the content-based messaging approach has proven to be the most flexible, adaptable and inclusive of the selected set.

The provision of decoupled communication, with simple client interfaces and an extensible message format for which schema can be defined, adapted, and shared between participants has meant that the majority of the development and runtime effort for participant processes is focused on each participants task at hand, i.e. term extraction and query composition for the Query Dispatchers, and query processing and result generation for the Link Resolvers.

6.2 Alternative models

This section examines four approaches for link resolution processes that are different from the database model of existing DLS implementations, with consideration to the findings of the previous section concerning query and result distribution where appropriate.

6.2.1 Tuplespaces I

Tuplespaces are shared, associatively addressed, loosely restricted bags of tuples, which are vectors of typed values. The tuplespace model was a result of work on the Linda coordination language of Gelernter (Gelernter, 1985; Carriero, 1989), which proposed a combination of a standard sequential language such as C with a small number of tuplespace manipulation primitives to produce a complete parallel programming language, e.g. C-Linda.

Tuples are associatively addressed in that they are retrieved by querying the tuplespace by matching partially grounded tuples (templates) with the tuples

present in the bag. Templates can include formal fields in the tuple that then only match tuples with fields of the same type in the same place of the vector, or they may be informal, where they would match if the values are equal, with a space-specific notion of cross-type value equality applying.

Tuple spaces provide a good starting point for distributed information systems in that they provision data communication, synchronisation, and a simple data repository all in one simple framework. Processes, irrespective of operating or coding platform, can inject tuples into a space for other processes to observe and then act upon. Key benefits include:

- Decoupled addressing. *No need for process to explicitly name the target of their message (tuple), or for the recipient to address the sender. Unless provisioned within the application, inter-application communications are anonymous.*
- Decoupled in space. *The associative tuple addressing approach imposes no structure on the nature or format of the tuples, and the lack of a visible tuple hierarchy means that the tuplespace is able to provide a globally shared space, regardless of machine of platform boundaries.*
- Decoupled in time. *Tuples survive applications that generate them, or may be interested in observing them.*

When compared with database systems, the lack of rigidity in the information model and the lack of support for bulk-update transactions or multi-item structured query casts tuplespaces into a kind of shared ephemeral communication buffer rather than a long-term data repository. Implementations of tuplespaces exist that provide a database (and thus persistent) back-end for systems that require that level of functionality.

Tuplespaces have been used as a service discovery and coordination mechanism, such as with Laura (Tolksford, 1993), which used tuplespaces as a brokering mechanism for service provider and clients, and as an object mobility technique in JavaSpaces (JavaSpaces, 2000), where arbitrary Java classes are communicated as tuples.

IBM's Almaden Research Centre has developed a Java-based tuple space system, TSpaces. A mature, openly available implementation, TSpaces comprises a TSpaces Server that manage the tuplespaces, and a lightweight client programming library (Wyckoff, 1998).

The motivation for TSpaces stems from the observation that *incompatibility* was the common stumbling block for applications that wish to transcend platform boundaries; that applications from personal computers to personal digital assistants, and automobile in-car computers to building atmospheric controllers, were becoming more connected in that networking technology meant that they had facility to communicate, but systems were largely incompatible with one another at an information level.

Being Java-based and with a small client library, TSpaces enjoys the same degree of platform independence as its implementation platform, and deployments have been demonstrated on handheld devices and laptops as well as larger, more powerful desktop computers.

TSpaces extends the primitives identified in Gelernter's Linda language to include additional data management features for persistence, indexing for querying; the ability to download both new data types and new server-side semantic functionality, provisioning real-time extensibility of the system; access controls to define what operations are valid on different components; and a trigger-based notification system providing clients with a mechanism to react to additions to the tuplespace, rather than having to poll continually.

The small operator set comprises blocking and non-blocking methods to insert, query for, and retrieve tuples from the shared spaces. Beyond the basic template-based structural matching offered by other Linda derivates as discussed above, TSpaces also includes the notions of typed tuple fields that observe an object-oriented inheritance hierarchy, named field matching and combinatorial query semantics.

One of the issues in any distributed information system is the careful design of the information space, especially where messages are 'anonymous' such as

with tuplespaces. Being based heavily on the Java object and typing system, the use of tuples, templates and the provision of partitioned spaces on the same server in TSpaces enables programmers to be quite precise in the design of their data space. Otherwise, the 'flatness' of the tuple model would mean that a popular space server would easily find clients injecting tuples that are matched by processes they were not intended for.

6.2.1.1 Architecture

One approach that would incorporate tuplespace technology in a pDLS link service would be to provide a TSpaces backend to individual Link Resolver processes, c.f. a database or table-based approach typically observed of link resolvers. The link model would therefore be internal to the TSpaces Link Resolver implementation and, so long as the query interface exported to the framework matched that expected by the other components, particularly the Query Dispatcher, no other component would be affected by this specific implementation of the local Link Resolver.

The TSpaces-backed approach might be chosen as a suitable Link Resolver implementation, for example, due to its template-based tuple matching ability. A space full of links asserted as tuples similar to those shown in Figure 6-13 can readily be matched by non-consuming scans that feature partially grounded tuples, e.g. Figure 6-14.

This is a favourable characteristic of the decoupled link service approach, and one might reasonably expect a plethora of different Link Resolver implementations that employ different back-ends all participating seamlessly in a pDLS system.

```
Link< '12', Anchor< '', '', 'hypertext' >, \
Anchor< 'http://www.hypertextkitchen.com/', '', '' >, \
Person< 'http://bill.local/concepts#mkt' >, \
Timestamp< '2002-08-02T03:11:01+0100' >, \
'Hypertext Resources' >
```

Figure 6-13: Example tuple representation of a link

However, this particular TSpaces-based experiment considers the deployment of tuplespaces as a unifying communications space for an entire pDLS framework that also enjoys a degree of persistence as to enable asynchronous and decoupled query. In this prototype, there is a single space server provisioned either as scenario-supporting infrastructure, or hosted by the first participant to join the scenario and determine that there is not a space server available.

The interaction modality with this architecture is that clients connect to the space server and select (or create, if not already present) the tuplespace on the server that corresponds to their user's node identity, e.g. `bill_local`. This choice of space partitioning means that tuples pertaining to a particular user's query will remain private to processes relevant to that user's view of the information space. The processes also bind to a framework-wide space where configuration parameters and associated service handles are mediated.

Query Dispatcher processes, upon determining a term on which links are to be resolved, assert a query tuple similar to that shown below. Each query tuple has a transaction-identifying element such that any links that result can be traced back to the query (and term) that instigated their assertion.

```
Query< Link< ?, \
Anchor< 'http://weed.local:6968/lesson.html', '674', 'hypertext' >, \
?, ?, ?, ? >, 'http://bill.local:6967/qd#1429'>
```

Figure 6-14: Example link resolution query tuple

Link Resolver processes employ the TSpaces notification service and therefore they are informed of the assertion of tuples that represent link resolution queries. In reaction to an appropriate change to the space, they invoke a non-consuming scan to retrieve copies of all of the query tuples that they are to resolve. Successful resolutions are then published as tuples back to the space, with appropriate transaction identifying elements similar to that shown here.

```
ResolvedLink< (contents of Figure 6-13) , \
'http://bill.local:6967/lr#example', \
'http://bill.local:6966/qd#1429'>
```

Figure 6-15: Example resolved link tuple

The Query Dispatcher process that originated the query can then consume the tuple from the space and include it in its result set. An alternative strategy that has demonstrated some benefit has been for the Query Dispatcher to perform a non-consuming scan of the tuples that result from Link Resolver processing (that is, to leave the resolved links in the space) such that other processes can see the results of previous queries, possibly beyond the lifetime of the source participants. A form of garbage collection is then applied (e.g. in result to a space utilisation threshold parameter, or as a tuple age parameter) such that residual resolutions are cleared up when they are sure to be of no meaningful use.

The mapping from links to tuples is a straightforward transform from the XML linkbase to hierarchical, typed tuple. The resolved link tuple example above is taken from section 5.1.4.

The linkbase identifier in this model is a participant element of each individual link. Other tuple definition patterns are possible without changing the query modality of the approach, for example using a single namespace for the entire pDLS framework, with user/node identifier being an explicit tuple element as well as linkbase identifier.

However, the additional hierarchy provided by segregating data spaces pertaining to a particular node's domain provides a suitable level of abstraction and, to an extent, protection from errant processes polluting other users' spaces.

6.2.1.2 Relative Merits

Reflected against the earlier prototype architectures, the following points are noted:

Pros

- Space/Time Decoupling
- Persistence
- Event Notification

Cons

- Scalability/Distribution
- Service Provision
- Source Identification

Space/Time Decoupling

Adopting Linda-style tuplespace models enable the heterogeneous devices and services in the scenarios of interest to be decoupled in both space and time: interaction irrespective of source of query in an asynchronous yet timely manner. Links, linkbases, document resources and service handles can be made available through pattern-based space interaction.

The tuplespaces also serve well as object caches, e.g. for partially resolved links that needed further resolution by different link services, or adapted content of individual documents. One subsidiary approach has been to implement resource bases as tuplespaces, where the resource metadata and either the resources themselves (for relatively small resources) or URLs to HTTP-accessible file stores are maintained as tuples in the appropriate space.

TSpaces is implemented in Java, and the only restriction on tuple fields is that they must be defined Java types, which means that arbitrarily large byte arrays (i.e. image files, PDF documents, etc.) *could* participate directly in the space as tuple fields. However, so doing would mean that for every instance that tuple matches a query, the entire resource would be communicated across the socket connection between query source and space server, which would be extremely inefficient for most uses.

Persistence

Queries and their results can persist in the space beyond the instant that they were asserted. This provides a decoupling in time that enables, for example, processes late to the scenario or busy at the time of original query to still be able to participate and enrich the space with their results.

It also provides for a degree of post hoc analysis of the component interactions for the purpose of evaluation, and, most beneficially, provides a means to preserve the hyperstructure resulting from the spontaneous presence of resources without explicit link or resource capture by user.

Event Notification

The facility for processes to register a query pattern (a partially grounded tuple) and then asynchronously be informed of the assertion of tuples that match that query provides for a model of 'reactive' processing, but with the additional benefit of persistence where the tuples 'hang about' until destructively consumed, for example un-asserted by the source, or removed as no longer relevant by a third party.

Scalability/Distribution

The TSpaces approach lacks scalability in that current implementations of the TSpaces server are confined to single nodes; spaces cannot be distributed over multiple space servers.

Service Provision

The lack of distribution means that spaces are restricted to single domain systems, for example within a meeting room, on a campus, but not Internet-wide.

It also means that there is no fault tolerance or load balancing available, resulting in a single point of failure and bottleneck for applications. However, the model, not the implementation, is under survey as a candidate

infrastructure approach for link service architectures, hence the applicability of the approach remains.

Source Identification

Similar to Elvin and IP Multicast as communication patterns, the source of a tuple in the space is not available as a term that can be queried for. Therefore, where that data is relevant, an application-layer solution is required. The prototypes developed use an explicit transaction identifier that serves for this purpose, e.g. matching results to queries. A potential issue that this introduces is that the identifying field can be faked by other processes asserting tuples with another processes' identity, and therefore the fidelity of the space compromised. However, in such closed scenarios this is not perceived as a serious issue.

6.2.2 Tuplespaces II

One of the limitations for the adoption of a tuplespace system was the lack of scalability in the current implementation with regards the distribution of space servers. Whilst an individual instance of a tuplespace server can be partitioned into individual spaces, and applications' foci scoped thereon, there is no provision for the distribution of tuples across different space servers, nor is there a mechanism to migrate tuples between different spaces within the framework.

One approach to combating this problem to provide distribution has been to produce a hybrid system comprising tuplespaces and a content-based message routing system. Different space servers maintain *local* tuplespaces for *local* data, and to employ Elvin as a mechanism for distributing operations between spaces, for example, through the definition of a notification schema that maps a subset of Linda operations to notification tuples.

This approach was developed from a co-incidental requirement for a mechanism to discover when devices with tuplespace services became present in the local network. Where no standard approach exists for the

discovery of a tuplespace server (clients typically bind to explicitly named server instances), an approach based on Elvin's anonymous, content-based messaging was developed. As TSpaces servers are started on new participant nodes that join the network, their presence is announced by virtue of an Elvin notification containing an identifier for the space that they serve (i.e. the node's name, and the fact that they are a TSpaces instance).

6.2.2.1 *Architecture*

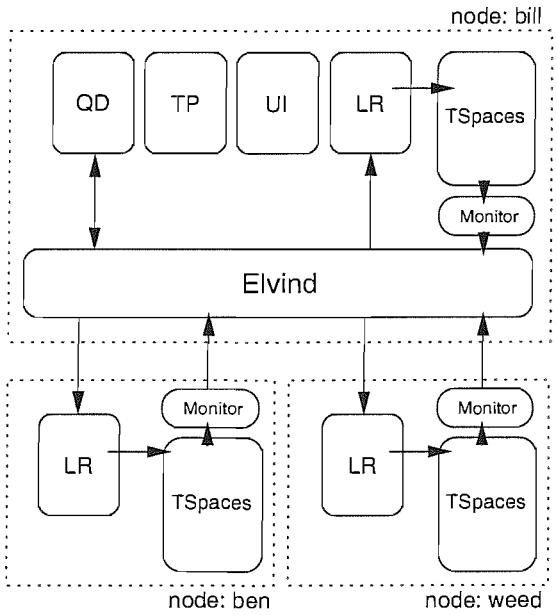

Figure 6-16: Distribution using a hybrid of TSpaces and Elvin

There are no changes to either the data model or the query interface compared to the previous tuplespace-based prototype, however query and update operations now need to be coordinated across multiple space servers.

In this architecture, processes monitor the local tuplespaces for the assertion of different kinds of tuple (e.g. link resolution query and results) and then mediate interactions across the different space servers. Elvin, in this instance, is a communications proxy binding together the different tuplespaces.

An example interaction in this case would be as follows (with reference to the example architecture above):

The Query Dispatcher process on node `bill.local` asserts a query tuple into the local space (`bill_local`). Having registered for the assertion of query tuples, the Space Monitor process on `bill.local` asserts an Elvin notification such that other, remote, Space Monitor processes are made aware of the query. This requires that the query tuple be encoded in an Elvin notification element.

A remote Space Monitor process receives the notification and asserts the encoded query in the local tuplespace subspace dedicated to interactions pertaining the remote note (i.e. space `bill_local` on node `ben.local`). Link Resolver processes on node `ben.local` then react to the query, process its contents and assert link resolution results back to the `bill_local` space on node `ben.local`.

The Space Monitor on node `ben.local` observes the assertion of results and, having appropriately encoded the tuples, publishes them as notifications. The Space Monitor process on node `bill.local` receives the notifications, decodes the link resolution results and asserts them as tuples in the `bill_local` space on the local node, `bill.local`.

The Query Dispatcher process observes the link resolution results sourced from the Link Resolver processes local to the node, augmented with results from the remote resources.

The mediation role in this prototype is performed by Space Monitor processes that act as the bridge between the local tuplespaces and the Elvin communications substrate. They subscribe to Elvin notifications concerning remote queries that require action by local resolver processes, and to remote results in response to locally originated queries. They also react using the TSpaces notification service to the assertion of tuples within the spaces that they monitor that require distribution to remote spaces for action.

Where the previous centralised TSpaces prototype featured an administrative space for housekeeping and configuration data, this distributed approach also

features localised administrative spaces, which are also mediated using Elvin as a substrate in a similar manner to the query interaction above.

6.2.2.2 *Relative Merits*

In comparison with the previous tuplespace-based infrastructure, the relative merits of the Elvin-mediated multiple TSpaces architecture are observed as follows.

Pros

- Failure Tolerance
- Data Location
- Inter-space Coordination

Cons

- Universal View
- Service Provision
- Loss-on-Leave

Failure Tolerance

If a node disappears, the only data lost to the scenario is the data and services that were hosted on that node, whereas previously, if the node hosting the space server left then the entire system failed.

Data Location

Likewise, data pertaining a particular node's interests is located, managed and mediated by local processes, more fitting with the ownership and management issues highlighted by earlier scenario analysis.

Inter-space Coordination

The use of Elvin as a mediating glue realises all of the benefits of the notification service approach as identified in section 6.1.3.2, but now with the additional benefit of persistence afforded by the tuplespace model.

Universal View

With a single, centralised tuplespace server, all tuples were immediately available in a single operation. Whilst this is still the case from a client process perspective (the Elvin inter-space coordination is hidden from client processes), there is a significant overhead imposed in attempting to coordinate between tuplespaces.

The example query interaction described for this prototype demonstrates that inter-space interaction can be straightforward when the level of coordination is quite coarse, however, in order to satisfy a more general query (e.g. to provide a uniform view of all of the tuples that match a particular pattern in all '`bill_local`' spaces) significant additional coordination is required.

Service Provision

Whilst gaining the benefits of an underlying notification framework, this prototype also inherits the issue of requiring an Elvin message router be present on the local network.

Loss-on-Leave

Whilst the issue of partial failure due to single node departure has been improved over the previous prototype, there is no direct mechanism for the hyperstructure (or the linkbases or resources) to survive a node's departure. The feature of local data being owned by local nodes also introduces an issue in that when that node leaves, its information is lost unless already discovered and captured by other participants.

6.2.3 Tuplespaces III

Lime, Linda in Mobile Environment (Picco, 1999; Murphy, 2001) breaks free from the centralised, client-server, monolithic tuple space model of systems such as TSpaces or JavaSpaces by offering Linda-like coordination across a distributed network of participants.

The model employed by Lime is that agent processes execute on participant nodes with their own local tuplespace whose members can be exchanged between processes using the Linda set of operators. The collection of spaces distributed throughout the system comprises a transiently shared or federated tuplespace in a manner that is transparent as far as user processes are concerned.

Participant nodes in a Lime environment provide execution platforms for the agent processes such that agents and their associated tuplespaces can migrate between nodes.

As each node maintains its own tuple set, should that node disappear, only its data is 'lost' to the federated space and not the entire systems, as would have been the case had the singular TSpaces service failed in an earlier prototype. The mobility provision means that, should the node departure be predictable (e.g. user signals intention to quit, rather than network or node simply failing), then the Lime agent and all its associated tuple data can migrate to a remote platform and survive the departure.

6.2.3.1 Architecture

The architecture is not too dissimilar from the earlier TSpaces-based prototype, except that rather than maintaining connections to a centralised space server, the Linda coordination operations are enacted on the local underlying Lime platform on which the individual pDLS processes are built upon, and the inter-node mediation (as provided in the previous prototype by Elvin) is now a feature of the tuplespace middleware itself.

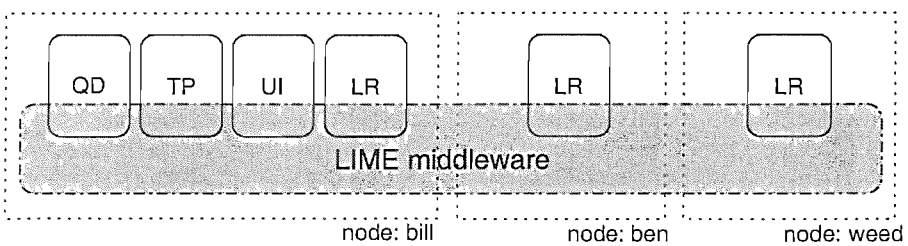

Figure 6-17: Distribution using Lime

The interaction modality is as per section 6.2.1.1 above, except that all tuplespace operations are performed against a local tuplespace service interface as opposed to a 'discovered' single centralised one. The Lime middleware marshals queries and collects results from the individual, distributed tuplespaces throughout the network automatically, providing a universal view abstracted away from the location of individual tuples' host platforms.

Aside from the abstraction of tuple location and the presentation of a unified space that affords, the major additional feature realised in this approach is that processes (agents, in Lime terminology) can migrate to execution platforms on different nodes of the network, for example, to reduce the network cost of distributed query satisfaction, or to provide a degree of survivability of their owning node's data once that node has expressed a desire to depart the scenario.

6.2.3.2 *Relative Merits*

These observations are made against the earlier hybrid TSpaces/Elvin prototype where benefits and failings differ.

Pros

- Distribution model
- Loss on Leave
- Linkbase Fragment Mobility

Cons

- Latency
- Fault tolerance and Subspace Engagement

Distribution Model

Like in the hybrid solution above, data is kept local to nodes that own it. However, with the Lime approach there is provision for data to be migrated within the framework closer to another process should it wish to perform

queries of a high level of intensity, or in anticipation of the participant node departing where contained data is desired to still be retrievable.

Loss on Leave

The issue of resource tuples and the processes that generate them being lost when a node leaves the scenario where it would otherwise be desirable for those resources to remain present can be catered for using Lime's agent mobility feature. Whilst enabling resources to continue participating beyond the departure of a participant, the issue of resource and process ownership becomes more challenging. It is unclear, for example, what should happen if the node to which the (now absent) user's processes have migrated then declares that it desires to leave the scenario.

The issue of ownership and management was catered for in previous examples in that, as identified by the scenario analysis, the node that owns the data, processes the data, providing a clear and readily identifiable authority for when something goes awry.

Linkbase Fragment Mobility

This infrastructure approach has been the first to afford a degree of mobility for processes, whilst still maintaining a single unified view of the shared communication space.

An additional performative is enabled in the Link Resolver's control interface (represented in the pDLS GUI) that enables a user to push their resolver instance to a different node in the framework, e.g. to improve the network utilisation and reduce repetitive query latencies.

Latency

Whilst conceptually a 'clean' abstraction separates the location of tuples away from the individual node of a distribute space on which they are currently asserted, the observed query and interaction latencies for scan and read

operations have been comparable if not worse than with the TSpaces/Elvin hybridised approach above.

The TSpaces/Elvin approach provided a degree of explicit control over which tuples were communicated across the network and at what time. The Lime approach has no such control, although its current use as an ephemeral communications buffer does not suggest that this is a major concern.

However, a tuplespaces-based approach where all resource data were instantiated as tuples in addition to (or instead of) live query and result data might suggest otherwise.

Fault tolerance and Subspace Engagement

Existing implementations of Lime, its underlying tuplespace model LighTS and the μ-code mobile agent framework cannot recover from unanticipated node departure. Likewise, existing implementations suffer an issue with the mechanism by which Lime agent processes engage with other agent processes in order to process queries across the entire distributed space. They do not currently take into account the presence of other Lime services that may be active on the local network. As a result, different Lime applications disrupt each other when coincident on the same network.

These are implementation issues with the existing Lime framework, though, and should not be taken as a criticism of the model of the approach.

6.2.4 Directory Services I

Directory Services are services whose sole purpose is to provide information about people and resources to clients requesting information. A trivial example would be British Telecom's telephone service, 'Directory Enquiries', where a client calls an operator asking for the telephone number of somebody, giving their name and address. The operator searches the directory database and returns the telephone number.

Internet Directory services are designed for slow moving data to be queried in a lightweight, timely manner. Their use, for example, in e-commerce systems is primarily as an integration service, providing a single point of information source for data about people, services, configuration, etc. – anything that has some loose structure and might be required to be retrieved at some point. Most implementations of the database back-end of a Directory Service are optimised for retrieval to the extent that directory updates (and therefore changes to the underlying database) can be prohibitively expensive.

LDAP, the Lightweight Directory Access Protocol (Yeong, 1995), is a standardised implementation of this type of service for the Internet. 'Lightweight' here signifies that its implementation is somewhat less complex and draconian as earlier protocols on which it is based, such as the ISO X.500 Directory Access Protocol (DAP), with less complex encoding rules, and a protocol binding that is open and readily available over a TCP/IP network stack. X.500 Directory Services typically require OSI network stacks, which are costly and by no means as ubiquitous as IP has become.

Various programming interfaces exist for LDAP, including C, Perl, Python, Lisp and Java language bindings, making it readily accessible to develop for.

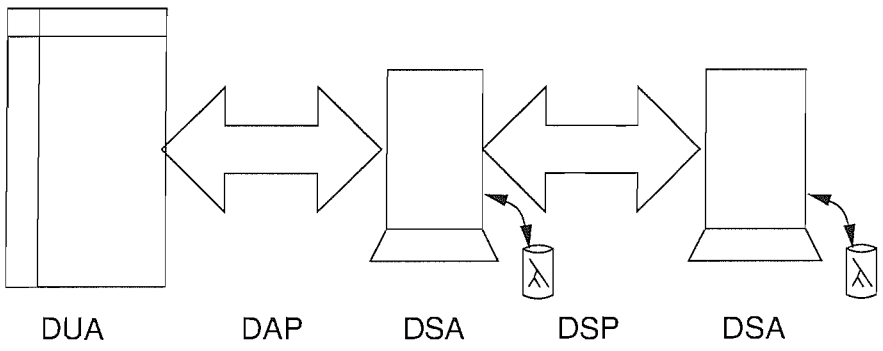

Figure 6-18: LDAP component model

The model of a distributed system employing LDAP Directory Services is shown in Figure 6-18. The model comprises the Directory User Agent (DUA) processes, such as the Mozilla browsers address book utility, which speak a Directory Access Protocol (DAP, here LDAP) to query a Directory Service Agent (DSA). The DSA then either responds directly to the query should it

have the necessary knowledge in a local database, or can instead originate a referral that may involve communication with another DSA using a inter-Directory Service Protocol (DSP). With the LDAP approach to Directory Services, both the DAP and the DSP are the same protocol, LDAP.

With version 3 of LDAP, DSAs can either instruct the client that it should query another server (client-referral), or it can itself query a remote DSA on behalf of the client (server-referral). The latter requires less complexity in client applications, but does introduce additional overhead for the DSA.

The use of LDAP as an instance of a Directory Service means that a unified naming scheme and standardised (simple) transport and querying interfaces are available. LDAP Directory Services can best be described by separating its four models:

- Information Model. *Defining how entries (entities) are organised within the Directory*
- Naming Model. *Defining how entities are referenced*
- Functional Model. *Defining the operations that can be performed on the Directory*
- Security Model. *Defining how different entities can be secured from different operations*

The Information model defines the basic unit of information within the directory as an *entity*, named by a distinguished name (DN). Entities are collections of typed *attributes*, where attributes may be specified as mutli-value (e.g. a person entity might have more than one telephone number). The set of permissible attributes is specified by directory *schema*.

The Naming model defines the hierarchy of entities within the directory to be an inverted tree, with a conceptual (but typically un-named and otherwise empty) root. This hierarchy is often referred to as the Directory Information Tree, or DIT. Comparing the LDAP naming model with a file system, there are many similarities, but three significant differences: The root is conceptual;

nodes of the tree are named leaf-to-root; and each node in the tree can be both a container of other nodes (c.f. a folder), and of attributes (c.f. a file).

The Functional model defines three groupings of operations for interacting with directories. Retrieval is realised through *Search* operations that return sets of entities, or *Compare* that confirms or not the presence of entities that satisfy queries without actually transporting the set of applicable entities to the client. There are four update operations: *Add, Delete, Modify* and *ModifyDN (Rename)*, which offer mechanisms to add or delete entities, change attributes within entities, or move the entity to another location in the DIT. There are also three authentication and control operations, providing mechanisms through which credentials can be provided to authorise access to subsequent operations, or to assist the directory server by reporting when the client is no longer interested in receiving any further results from a query.

The Security model defines mechanisms to control which operations are permissible on which entities, including access to individual attributes within entities. The Security model is the least defined of the four, and various different LDAP Directory service implementations have their own approach to securing the DIT. For example, OpenLDAP uses server configuration file directives to control access at start-up time, where SunONE (formerly iPlanet Directory Server) uses specific attributes in the directory entities themselves.

For reasons of performance, reliability and scalability, recent iterations of the LDAP specification have introduced the notion of DIT replication, where different parts of the tree can be replicated to different, distributed servers. This provides a mechanism for increased performance in that more frequently queried data can be moved to services closer to the sources of the queries. It aids load-balancing of servers and therefore reliability as more than one server is available to satisfy the same query, and also facilitates local data ownership in that data can be 'writable' at one location (i.e. an authoritative site) and then read-only copies replicated out to other servers.

LDAP was first applied to link service infrastructure in (De Roure, 2000). The work examined LDAP Directory Services and Whois++ Query Routing as

alternative technologies to an HTTP-based transport for link services distributed throughout a spectrum of connectivity beyond intranets, where the DLS had typically been deployed. Where they considered intermittent connectivity as a constraining characteristic of their target deployments, they did not actually exercise that notion in their experiments.

They surmised that a Directory Services approach to wide area distribution of link services was suitable if slow given existing implementations available at the time, and that the query routing model of Whois++ was an appropriate model when forward knowledge regarding the specifics of remote services was available.

However, the nature of forward knowledge was observed as being difficult in the fixed network topologies investigated by (De Roure, 2000), and the likelihood of such summary knowledge being available for distribution within a spontaneous system as proposed by the pDLS activity here renders the Whois++ query routing approach inappropriate. Their criticisms of Directory Services, particularly the lack of support for within-DIT referral between service instances, are largely catered for by recent Directory Service implementations.

The considered adoption of LDAP directory services within the pDLS comes about due to the unified model of the Directory Information Tree in combination with the flexible query semantics and sub-DIT referral provision of modern LDAP server implementations.

6.2.4.1 Architecture

Where discussion in section 6.2.1.1 regarding the use of tuplespaces observed that particular technology could be 'hidden' behind a Link Resolver query interface without impact on other processes in the pDLS, the same strategy could be taken with LDAP. However, the approach with this prototype intends to make use of the unifying model of a scenario-wide directory information tree, and as such the 'hiding' proposal is moot.

This first Directory Service-based prototype employs a single scenario-wide Directory Service Agent and the model taken is that clients register all of their link data in the DIT using the naming model proposed in Figure 6-19. The schema for link data extends the standard LDAP information model to include notions of linking as defined in section 5.1.2 above.

The DIT is structure obeying the recommended model whereby containers within the tree are collected by node, and that linkbases are then discrete sub-containers thereof. For example:

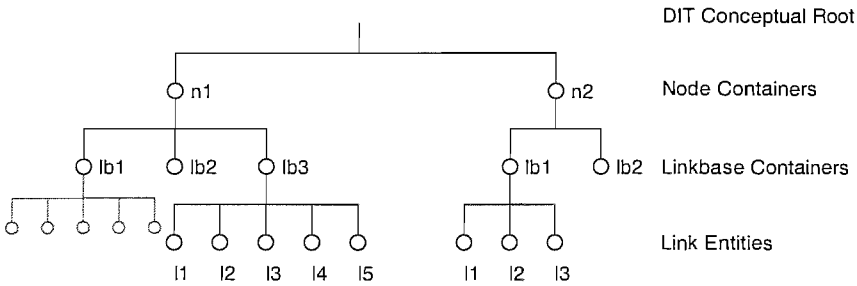

Figure 6-19: LDAP DIT for unified information space

There is no change to the conceptual link model as is exemplified by the fact that an XML Stylesheet Transform (Clark, 1999) can be applied to XML linkbase data to convert it into LDIF (LDAP Data Interchange Format) for injection into the Directory Information Tree.

```
dn: lid=l2, lb=example_linkbase, node=bill
lid: l2
sourceSel: hypertext
destResource: http://www.hypertextkitchen.com/
timeStamp: 2002-08-02T03:11:01+0100
owner: uid=mkt,node=bill
title: Hypertext Resources
objectclass: link
```

Figure 6-20: LDAP Directory Interchange Format example link

With the DIT populated, a typical query interaction is for the pDLS query component to query the DIT using sub-tree search algebra of the LDAP functional model, rooted at the linkbase branches configured as enabled linkbases by their directing user. In this centralised example, the DIT is the single source of all knowledge, and therefore only links that are entities within the DIT are discovered as a result of query.

6.2.4.2 Relative Merits

Pros

- Model applicability

Cons

- Static data
- Provision, Discovery

Model applicability

The DIT model's hierarchy fits well with the hierarchical nature of the hyperstructure information in the scenario spaces. Collecting linkbases under nodes from which they originate is sensible and provides a clean mechanism for identifying the source of resources. Combined with a flexible and powerful query model, the use of LDAP as a link service information model has been shown to be entirely appropriate.

The hierarchy achieved with the DIT and the functional model of LDAP means that, should a linkbase or other entity captured or reflected in the DIT migrate to a different node, a ModifyDN operation would mirror the change within the directory. Likewise, as copies of resources are made, directory entries can also be copied, or sub-tree referrals used so that the DIT representation matches the location of the data that it represents.

Static data

There is no facility for processes to be invoked or informed when a particular query is performed of the DIT. Therefore the dynamic generation of links in response to queries is no longer possible. Data within the DIT is static, by intention of the typical nature of the service.

Provision, Discovery

With this single DSA approach, there is a need to employ a third party discovery service for user agent processes to discover the Directory Service in order to be able to connect to it. DNS-SD has been deployed on Internet-scale networks for this very purpose, and as with the other centralised services above, DNS-SD applied in mDNS zeroconf networks has been demonstrated as successful.

Most off the shelf LDAP implementations, such as Sun ONE Directory and OpenLDAP, implement a degree of fault tolerance, load balancing and reliability by offering failover and replicated services, either clustered in local networks, or distributed across wide areas. However, these approaches are not feasible in the local area scenarios of interest, primarily due to the need for manual, static configuration of peer services, but also to the mechanism by which replication is achieved.

DIT replication is an out-of-band service that, taking OpenLDAP as an example, directly manipulates the Directory Service's back-end database to intermittently reflect changes out to replica servers. Whilst this works well on Internet-scale directory deployments, and is in-keeping with the design goal of directory services being targeted at 'slow-moving' data, it offers poor utility to the local, more dynamic and fluidic environment of the candidate scenarios.

6.2.5 Directory Services II

A primary failing of the previous Directory Services-based experiment was the lack of support for dynamic or computed links. A similar observation can

be made of the use of tuplespaces as linkbases, and this is a core motivation for the abstraction of link service implementations behind a unified query interface.

This second LDAP-based experiment considers a centralised server instance as a shared result cache for link and resource queries, in a similar manner to the tuplespace experiment of section 6.2.1. Whereas in the equivalent tuplespace experiment, query mediation was achieved through the assertion and consumption of tuples, the only interaction for processes with the directory service in this approach is the publication and subsequent retrieval of results. The query instigation is achieved through third party means. For the purposes of this experiment, that mechanism is the Elvin notification service, having been identified as a suitable communications substrate.

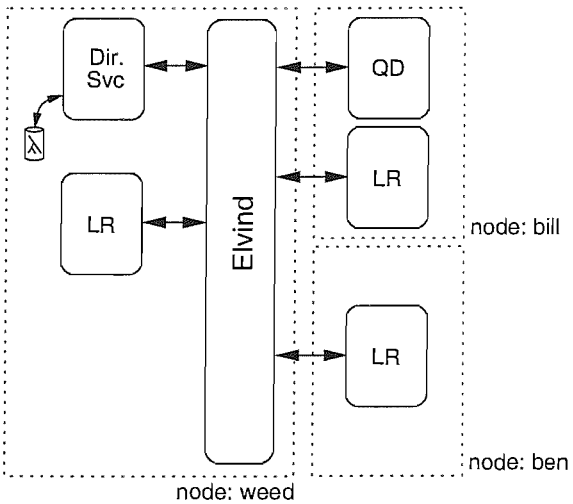

Figure 6-21: Distribution using single DSA and Elvin

The interaction modality with this architecture is that clients connect to the directory server and ensure that the container pertaining to their node exists in the DIT, creating it if not. Link Resolver processes that register also ensure that the container entities for their respective linkbases exist.

As with the Elvin prototype in section 6.1.3.2, when a user navigates, the Query Dispatcher publishes query notifications that all (matching) Link Resolvers then react to.

Successful resolutions are inserted into the DIT by the Link Resolvers as elements in the appropriate container, with appropriate transaction identifying attributes added similar to that shown here. The LDAP informational model supports the notion of multi-value attributes (c.f. the set-like behaviour of Elvin notifications). This means that should the link already have been resolved as a result of a previous query, the new query transaction identifier can be added as an additional attribute to the entity.

```
dn: lid=l2, lb=example_linkbase, node=bill
lid: l2
objectclass: link
objectclass: resolvedlink
sourceSel: hypertext
destResource: http://www.hypertextkitchen.com/
timeStamp: 2002-08-02T03:11:01+0100
owner: uid=mkt,node=bill
title: Hypertext Resources
resolvedFrom: http://bill.local:6967/lr#example
resolvedDueTo: http://ben.local:6966/lr#852
resolvedDueTo: http://bill.local:6966/lr#2112
```

Figure 6-22: Example Link resolved by two independent queries

The Query Dispatcher process that originated the query can then perform a sub-tree search of the DIT rooted at the containers relating to the linkbases that their user has expressed an interest in, to collect its result set.

6.2.5.1 Relative Merits

Relative to section 6.2.4, the applicability of the directory service model is reinforced by this approach, with the issues relating to centralised services requiring provisioning and third party discovery processes still hold.

Pros

- Static Data
- Opportunism

Cons

- Completion Signalling

Static Data

By modifying the use of the Directory Service to serve as a unifying result cache, it is once again possible to include Link Resolvers that compute link data in response to queries dynamically, rather than straightforward table lookup.

Opportunism

One benefit of the chosen DIT structure is that Link Resolver processes that have not explicitly been selected as enabled by the user can still subscribe to their resolution queries, process them against their local linkbase (or enact their computation process to generate dynamic links) and then publish the results to their own part of the DIT without interfering with the bona fide query interaction.

If the user were then to enable these linkbases, for example, before navigating to a different resource, then this additional result set could be made available immediately.

Completion Signalling

Having observed that asynchrony is a benefit for distributed processes to perform their own processing before returning results to a common pool, there is an issue pertaining to how long the results presentation should be deferred for.

In the examples presented, individual Link Resolution processes could signal their completion, e.g. by asserting a notification to that states no further resolution results will be forthcoming. However, given the anonymous nature of the communications substrate, the Query Dispatcher processes have no immediate facility for knowing how many Link Resolvers they are waiting

for. For instance, there might be more than one Link Resolver that responds to link resolution requests made of one labelled linkbase in the pDLS GUI.

The approach taken in these decoupled prototypes has been for the result collators to intermittently query the result caches for results, relying on the underlying query cache being optimised such that repeated instances of the same query on the same result set are cached and therefore not computationally expensive, except when new results are available for the result set. Within the bounds of a time-out threshold, the Query Dispatcher repeats the result query until its user navigates to a different resource, at which point the cycle starts over.

6.2.6 Directory Services III

The previous prototype successfully used the DIT as a scenario-persistent results cache, fragments of which could be downloaded by utilities and taken away from the scenario affording a level of persistence to the generated hyperstructure.

Modern LDAP server implementations offer different sub-tree referral mechanisms that aid in the distribution of the DIT throughout a network (typically Internet or enterprise scale, admittedly). The entity-attribute referral offers a 'see-also' semantics, typically observed in directory service implementations pertaining to people entities.

An example of this is a University directory where a single person would otherwise appear in two places in the DIT due to the naming model's design; a person that is both a student, and thus might appear in a student branch of the DIT, but also a member of staff, and thus also appear in the staff branch. Attribute-level referrals facilitate the data to be located in its appropriate container, whilst being able to reference other parts of the tree to service requests.

Another referral mechanism is the suffix referral, by which entire branches of the DIT can be distributed throughout different directory server instances for

ownership and query. This is different from multi-master replication, by which different portions of the DIT are authoritatively 'owned' by different servers and copies pushed out to replicas.

This third and final directory service prototype employs suffix referrals such that the DIT sub-trees pertaining to results from linkbases present on a particular node are physically located on directory service instances on those nodes.

6.2.6.1 *Architecture*

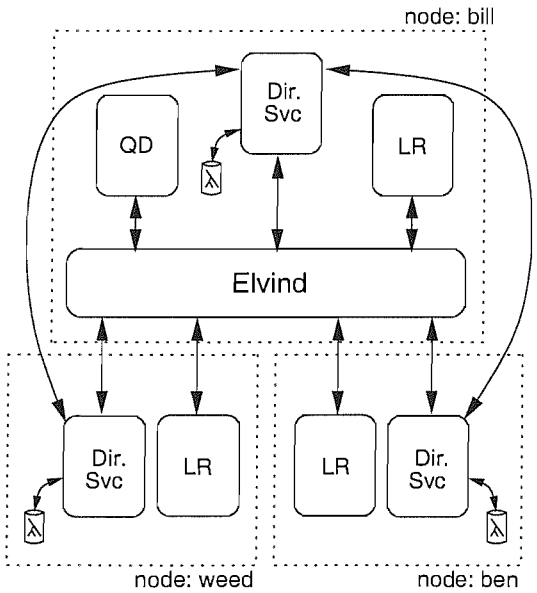

Figure 6-23: Distribution using multiple DSAs

The interaction modality is as per section 6.2.5, except that all directory service operations are performed against a local directory server as opposed to a single centralised one. The local directory server collects results from the individual, distributed directories transparently to the querying component, providing a universal view of the DIT abstracted away from the location of individual directory branch host platforms.

6.2.6.2 *Relative Merits*

In relation to the previous Directory Services-based prototype, the following relative merits are observed:

Pros

- Ownership

Neutral

- Discovery

Cons

- Persistence
- Configuration

Ownership

Link data resulting from queries in this approach are now cached local to the nodes that own the data. This is, however, at cost of persistence, as discussed below.

Discovery

Rather than having to employ a third party discovery service to find a centralised directory instance, client processes now only interact with their local service, and therefore the interaction is simplified.

However, there is still a need to discover other directory service instances within the framework in order to build the distributed-but-unified DIT.

Persistence

The centralised directory prototype provisioned persistence in that results cached in the DIT remained when nodes that generated them left the scenario. In this prototype, however, the sub-tree becomes unavailable as the node hosting them leaves. It is feasible for a process on the node to move the sub-tree to a different suffix using 'modifyDN' operations, and thus relocate the data onto a different node, however this coordination is nontrivial, and reasserts the issues regarding the subsequent node's later departure.

This is aside from the configuration issue arising from when the node leaves the scenario, where other nodes' configuration has to be updated to remove references to the now invalid suffix so as not to attempt query referrals to a non-existent service.

Configuration

Existing LDAP server implementations require that suffix referrals are a configuration-time directive, and cannot be changed without a server being shut down, reconfigured and then restarted.

Whilst an obstacle to deployment as a live service, the approach has been useful to explore the suitability of a distributed DIT as a results cache within the pDLS framework.

6.2.7 Multiple User Dialogue (MUD)

The first MUD was created by Richard Bartle and Roy Trubshaw at the University of Essex in 1979-80 (Bartle, 1990). Since then different kinds of MUDs have evolved, with various implementations. The MUD Frequently Asked Questions document states the following (Smith, 1990):

> A MUD (Multiple User Dimension, Multiple User Dungeon, or Multiple User Dialogue) is a computer program which users can log into and explore. Each user takes control of a computerised persona (or avatar, incarnation, character). You can walk around, chat with other characters, explore dangerous monster-infested areas, solve puzzles, and even create your very own rooms, descriptions and items.

Not all MUDs involve monster-infested arenas, or the 'combat features' that characterise some. Most MUDs share the same concepts and characteristics, however, in MUD terminology, the style of MUD environment discussed here is a derivative of LPMud named after the original author, Lars Pensjö.

The key characteristics that make MUDs an interesting and appropriate candidate approach for this research are:

- *Metaphor of real life.* In a MUD, people and things exist in a place, and people interact with their environment as they would in real life

- *Programming model.* The object-oriented, event-based programming model enables us to attach behaviours to virtual representations of real-world objects, and model the state changes on a per-object basis, for example as a reaction to an observed interaction

- *Extensible in real-time.* Not only the ability to create new locales and objects as the usage scenario requires, but also to adapt the functionality (the 'game logic') of existing interactions whilst the MUD is running, enabling dynamic and efficient modelling

- *Decomposition after the fact.* Through maintaining extensive event logs, different parts of the scenario can be analysed, or even replayed, virtually

In this MUD-based prototype, the motivation is to utilise the MUD objects as persistent containers in a similar manner to the tuplespace and directory service approaches previous. However, a key characteristic exercised in this prototype is the flexibility afforded by the fact that objects can be assigned scriptable behaviours, which suggests a mechanism for the inclusion of all link data, including that generated dynamically as a result of inquiry, within the model. Also, the MUD model's notion of containment relationships offers interesting properties to model the ownership and communication of link data.

6.2.7.1 Architecture

The MUD implementation chosen is a centralised server on which the virtual environment is built as the instantiation of 'objects'. These objects can be simple containers that act as data stores, or they can have behaviours scripted through which they can interact with other objects within the MUD.

The various components of direct interest from a pDLS prototype perspective are:

Rooms. 'Rooms' in this MUD are containers that represent the different participant nodes in the scenario, much like the channel mechanism used in the earlier IRC prototype. 'Exits', or links between rooms, are automatically created from each room to every other room for the purposes of inter-node navigation for any scripted processes built within the MUD.

Static and Resolved Link Objects. These are simple containers whose internal data model is a direct mapping from the XML link model as used in previous experiments.

Static Linkbase Objects. These are simple containers represented as an extension of the adopted MUD's 'box' artefact, in which Link objects can be collected. These are typically contained by other objects that transport collections of links, e.g. Linkbase Resolvers and the Query character below.

Static Linkbase Resolvers. These are NPCs (Non-Player Characters, 'beings') within the MUD with which other NPCs can interact, e.g. to query for Links. Their query code is, in the first instance, a simple local object lookup against the contained Static Linkbase objects, instigated by the 'resolve' performative being spoken by a querying character.

Dynamic Linkbase Resolvers. These are NPCs that exhibit the same characteristics and respond to the same command verbs as their Static counterparts, but rather than querying local collections of static links, they either compute any possible link resolutions on the fly encoded in MUD-code, or communicate the observed query with an external process, presenting any results back as above.

Query Dispatcher. The 'Dispatcher' in this prototype is typically an NPC that interacts with the other objects and characters of the virtual environment as a result of external stimuli, as demonstrated in the example interaction below.

However, its role can also be acted out as a Player Character (e.g. a user, or an administrator) and the virtual environment explored and manipulated directly from within the game engine.

When a node joins the scenario, it registers its room code with the MUD and invokes a process that knits that room object in with exits to and from other nodes in the model. The various pDLS component processes that have representations in the MUD also upload their 'game' code and instantiate themselves in their respective 'rooms'. For example, any static linkbases are published as Link objects contained by a mediating Static Link Resolver NPC, which is realised in the room.

Likewise the NPC instances are created that represent computed link resolvers, e.g. as proxies to external query processors backed for example by a tuplespaces-based link resolver, or as link resolution game code that computes any destination links based purely on data available within the MUD environment (e.g. links to resources on other nodes as part of the live metadata resource catalogues).

With the environment primed, the Query Dispatcher NPC joins the MUD. The pDLS query modality is such that when the user of the system navigates to a resource in their browser, their representation in the MUD (the Query Dispatcher NPC) enacts the 'resolve' verb on each resolver NPC that is selected as enabled by the user in the pDLS GUI.

Each resolver NPC then creates (or clones, in the case of the static linkbases) Link objects that are successful resolutions of the query, and places them in at least two containers: the room's 'Resolved Links' box object, which acts as a scenario-persistent store for successful resolutions; and the invoking Query Dispatcher's 'Resolved Links' sack, that is, somewhere within the containment hierarchy of the querying NPC.

An analogous interaction found in many role-playing MUDs would be that of a player visiting a merchant's store and purchasing wares. The player enters

the store, requests something, and the merchant puts it in the player's bag for them to take away, only here, the goods are free!

6.2.7.2 *Relative Merits*

This approach suffers similar issues to the other centralised systems approaches discussed earlier. However, the following observations are made in addition:

Pros

- Dynamism
- Persistence
- Component Discovery
- Interaction Model
- Debug

Cons

- Distribution

Dynamism

The MUD object model offers a very high degree of flexibility and dynamism for artefacts. For instance, the ability to have both objects that can be treated as nomadic data and objects whose behaviour can be scripted as processes that react to the nature of the query abstracted behind the same interface. As a result, the MUD approach is the first of those examined within the pDLS framework that enables the integration of link data itself and processes that generate it into the infrastructure, as opposed to just the integration of processes that examine or generate data as a side effect of operations within the infrastructure.

Persistence

The objects in the MUD exist independently of any external processes. Links, linkbases, and resolvers whose code is pure MUD object code can all survive the departure of the node to which they relate.

Likewise, the room in the MUD that represents the node can remain, although the coupling then between the real environment and that modelled by the MUD is diluted.

Component Discovery

Once represented in the MUD, the interfaces to the various services modelled can easily be discovered, whether through navigating nodes explicitly and requesting container inventories (e.g. perform 'look' in a room, 'examine' on a linkbase), or by directly traversing the containment model of the underlying game code (e.g. game environment search for the resolver character that contains the linkbase whose description attribute is 'iihf_rules' in the room 'bill_local')

Interaction Model

The mode of interaction observed between objects and characters in the MUD is familiar to those in the real world, which stands to reason as part of the design goals of other MUD systems. Giving a link or linkbase to another person has an accompanying digital counterpart action, the meaning of which is consistent between the two worlds.

Debug

The Query Dispatcher NPC role can be acted out directly as a 'normal' player would connect to a gaming MUD, wander round the nodes, discover artefacts (such as resources, links and linkbases), interact with processes, and even externalise linkbases for integration into a different DLS implementation. An invaluable debugging and monitoring tool, this interface is also a viable end-user interface.

Distribution

Whilst acknowledged as a characteristic of all the centralised component approaches investigated, the MUD approach in particular readily avails itself of distribution. The choice of MUD driver for this prototype, however, does

not. As a tool to explore the affordances of the MUD modelling approach, and to examine the appropriateness of the interaction model, however, the chosen environment has sufficed.

6.3 Experiment Summary

The previous sections have presented a number of computing models and experiments that explore different aspects concerning the provision of link services within the pDLS framework of the previous chapter.

The methodology employed by the experimentation phase of this research meant that the findings of earlier experiments informed the design of later instances, meaning that a tabular cross-comparison of approaches to infrastructure composition is not appropriate; areas of clear merit are readily identified and demerits considered as constraining factors for later experiments.

First, we introduced a decoupling of the DLS into its constituent functional components: the User Interface, Transport Proxy, Query Dispatcher and Link Resolver. The Transport Proxy performs resource retrieval on behalf of the user and passes a copy of navigated resources to the Query Dispatcher for analysis. Depending on the strategy employed, the Query Dispatcher performs term extraction to generate link resolution requests that it then distributes to Link Resolvers that have been enabled, aggregating their results before presenting to the User Interface.

This decoupling, itself novel for DLS implementations, has been shown to enable the impromptu combination of components co-incidentally available in an environment to be utilised in a familiar and readily available manner to provide resource discovery, inter-navigation, and include participation of physical artefacts and information resources in a live and evolving hyperstructure.

Eleven prototyping experiments are documented that have exercised different communications and computing models applied to the decoupling of pDLS

components. For brevity, the interactions with the utility components as discussed in the previous chapter are not shown.

The first four experiments consider different communications models, commencing with a one-to-one explicitly managed pattern through to an address-decoupled one-to-many pattern that exhibits great flexibility and applicability. The facility for initiating processes to 'fire and forget' messages, and for consumers of the information to declare an interest in patterns of messages without explicit management or large overhead renders the content-based routing publication-subscription approach – such as that availed by the Elvin system – favourable.

Taking the communication pattern experiments first, the decoupling of functional parts compared to monolithic DLSes enables differing degrees of participation by nodes that may not have a full suite of pDLS utilities available, or may be resource constrained for other reasons (e.g. connectivity, processing power, user interface issues).

The distillation of functionality into utility components (e.g. resource bases, anchor transmogrifiers, etc.) and core pDLS components providing bare necessities for link resolution, content transport and inter-component query dispatch is particularly advantageous in that new functional components can be added or removed from the infrastructure independently of those components already available extending the functionality of the service to the users, for example the development of a GPS Transmogrifier that adds a different kind of physical anchor and resolution component to the existing suite.

As regards the communication patterns, the IP and application-level multicast approaches to the distributed-component pDLSes were more resource efficient than the unicast approach to distribution.

Employing a session-based semantics to the otherwise connection-less IP multicast approach still falls short of the benefits realisable from the content-

based routing, and the bus/channel approach demonstrated to be less flexible than content-based messaging.

The second series of experiments introduce the notion of an ephemeral results cache as a 'shared state' across all of the various pDLS components, on each of the participants in the system.

The tuplespaces experiments provide a flat space in which data can be asserted and retrieved asynchronously by interested recipients through a template-matching technique, whereas the Directory Services prototypes offer hierarchy (of which a node-based location hierarchy was explored) and a more flexible, expression-based query interface.

Implementation technologies aside, both forms have similar merits, with the tuplespaces approach proving to be more flexible in information architecture and most readily adaptable to the dynamic nature of the chosen scenarios.

The final experiment that exercised a virtual environment model allowed a great degree of flexibility and enabled a direct interaction modelling approach through which familiar interactions (giving and receiving of links, for example) held similar semantics in the virtual model as they do in the physical world.

The sensing and enactment of physical-digital interactions of interest with scriptable orchestration of those interactions digitally within the context of a hypermedia information system was an interesting extension to the approach of a component-based infrastructure. The experiment validated the earlier approaches as regards decoupling of components in space and time, but also reinforced the notions of local ownership of data and of 'clustered' co-operation in the sense of local interactions between local participants being kept separate from those interactions elsewhere.

Through the series of experiments involving the development of the utilities to demonstrate applicability of the idea of distillation of DLS function into distributed components and the different distribution patterns and

computational models employed for infrastructure component interaction, experience suggests that the required core characteristics of a pDLS framework are as follows:

Decoupled in space. Data (Resources, Linkbases) and processes can exist independent of location; that interaction between processes can be contextual such that the right processes interact by subscription and address-less notification rather than explicit engagement;

Decoupled in time. That asynchrony amongst independent co-operating processes affords utility, particularly where synchronous interaction would impede delivery of core data (i.e. the resources navigated) to the users; that resolution results be cacheable and able to survive the temporary or permanent loss of a participant component, with an awareness of the loss of fidelity and provenance;

Considering these two core characteristics as sacrosanct, the most appropriate technologies within the context of the assumptions made about the environment and the scenarios defined as of interest are a content-based routing communications infrastructure employing notification and subscription combined with a type-free data co-ordination language for the conveyance of hypermedia link data, link resolution queries and results, and other auxiliary data.

Figure 5-5, our initial pDLS architecture diagram, can be recast in light of these recommendations, and is shown overleaf as a synthesis of the original figure and the aspects of the eleven experiments of this chapter.

The figure shows four participant nodes: One 'fully functional' node, **`bill`**, has an instance of the pDLS Core suite of components comprising proxy transport, query despatch and link resolver all fronted by a pDLS user interface (Figures Figure 5-4 and Figure 6-2). **`bill`** also shows two utility components loaded: an IRC bot used as a linkbase and a resource base comprising that node's local resources.

The figure shows two 'reduced functionality' nodes, **ben** and **weed**, that have no pDLS core instances, but do have utilities to participate in the scenario (a linkbase resolver and a resource base respectively). The final participant, **tablerfid**, has a link resolver and an RFID tag reader plus relevant physical transmogrifier utility process enabling the detection, resolution and therefore participation of physical anchors (tagged objects) in the information space.

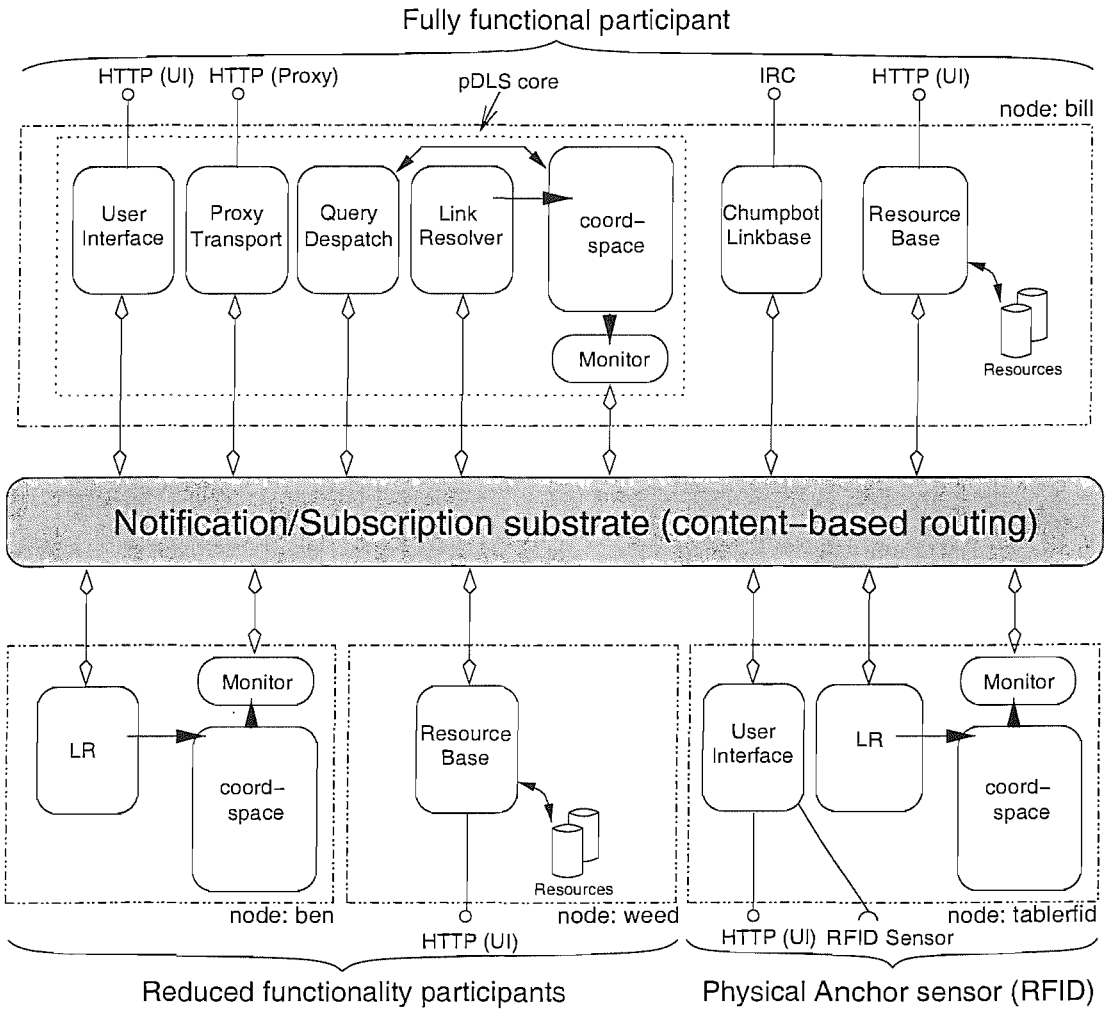

Figure 6-24: pDLS architecture, revised

The next chapter concludes this thesis by reflecting on the approach taken and its findings, and highlights on-going and future areas for consideration arising from the activities of developing the pDLS.

Chapter 7 Conclusions and Further Considerations

The central motivation for this thesis has been the desire to extend the affordances of hypermedia-enriched information spaces from their current domain of pre-determined, pre-configured enterprise networks into domains where the arrangement of information, services and users are less organised, local, and spontaneously available.

This idea has been supported by observations regarding trends in information systems usage, and accompanying trends in pervasive and ubiquitous computing, towards a world of available networked devices.

To scope the work, three candidate scenarios were identified that served to form a set of requirements for the information service, and the Distributed Link Service (DLS) approach to Open Hypermedia noted as an appropriate basis for a solution.

The approach taken in identifying a framework of cooperating components has been demonstrated as a successful strategy. The assumption asserted in section 4.5.6 regarding the nature of the underlying network infrastructure, and the adoption of a quasi-transparent approach to content monitoring (i.e. the transport proxy) has enabled different framework components access to each other and, unobtrusively to the user, access to the spontaneously available information as users navigate the space.

Eleven prototype implementations have been developed that investigate different computing models for the infrastructure, primarily focused on the facilitation of link services, distributed resolution a principal goal. Many of the negative observations made focus around artefacts of employed third-party technology implementing different aspects of the models, rather than criticisms of the models themselves.

Whilst the various prototyping exercises have resulted in the framework approach being validated as an appropriate technique for realising the requirements of the scenarios, and therefore serving as self-evaluation technique, there has been no user-based evaluation of the utility of the systems developed. The determination as to whether that system then has a perceived utility to its users is an additional item for future research.

7.1 Reflecting on the Architecture Experiments

The key difference between the two sets of architecture experiments in sections 6.1 and 6.2 is that the latter feature the inclusion of a link resolution results cache into the infrastructure, provisioning decoupling in time between query source, result generation, and result manipulation. This decoupling enabled an asynchronous interaction style between components in the framework that has demonstrated great benefit, for example, when nodes are temporarily disconnected from the framework.

It has also provided a mechanism for further post-query resolution result utilisation by other processes. An example is when a user, having navigated to a resource and therefore had links made available based on the configured resolvers and linkbases available at the time of query, could then launch an additional utility that was not running at the time of the query to capture the resulting hyperstructure; or perform some task such as link composition across the set of available links in an attempt to discover alternate representations or versions of link targets that may have become available since the query was enacted.

The basis of providing results caches instead of a linkbase-unifying publication store enabled the inclusion of link resolvers that generated links as a result of computation, rather than the typical lookup-and-match modality observed of traditional DLS resolvers. Of the different models investigated, only the MUD model enabled link resolution code to be written as a component of the framework vis-à-vis the other models in which external processes monitor a shared communication buffer. Attempts to achieve in-component link resolution in the other models, e.g. link resolution as a tuple

matching process or as an LDAP sub-tree query, would not enable the computation of dynamic links; the linkbases would all be static.

Latency was always going to be an issue when the process of data query is adapted from in-process in-memory lookup to a sockets based distribution. However, the nature of the local network has meant that the transport latency observed has been far less than in other distributed DLS efforts where Internet traversal is required. With link resolution delivery occurring asynchronously to the user's resource navigation query, the navigated resource is analysed out-of-band and thus the latencies of distributed link resolution hidden from view.

The content-based message routing model of the experiment in section 6.1.3.2 has proven to be extremely flexible. Once a dictionary or schema of attributes had been agreed for all communication between framework components, the ability to 'fire and forget' notifications without senders explicitly naming or maintaining lists of recipients has been a benefit. This approach has been readily adaptable such that additional components can be added with no overhead for or impact on other components in the framework. Likewise, the ability to 'fake' unicast message delivery by explicitly matching notifications to subscriptions that are assured (within the agreed schema) to match only one recipient has provided an in-band mechanism for, e.g., process command and control.

The notion of Access Control as expressed in sections 4.6.1.4 and 5.2.7.4 has been under-utilised in the developed prototypes. In part because the research questions have been primarily concerned with facilitating access rather than controlling or restricting it, but also because distributed security a difficult problem in its own right. Further consideration to a more refined notion of access control than the 'private or public' facility currently supported would most likely be grounded on the notion semantic mark up of resources and associations, informed by Semantic Web activities such as the Friend of a Friend and Web of Trust vocabularies (Brickley, 2003).

Concordantly, issues regarding digital rights management in general, in particular micropayments, copyright, and *trans*-copyright (in systems that permit Nelson-like transclusion) that have not been addressed by this facilitative work serves as an area of further interest. For example, consideration of how different access restrictions can be applied depending on the context of access required (e.g. reading versus annotation versus update) and the role or level of trust of the entity requesting the action, possibly derived from the community of practice-inspired model as an extension of the Friend-of-a-Friend model.

A Friend-of-a-Friend approach to access control assertion would include expressing relationships of trust between participants in RDF, and then expressing permissions as statements regarding those relations. The inference across asserted facts regarding resource access may simply be a matching process, akin to hypermedia link resolution. Extending that notion to other resource metadata and building on the hypermedia link resolution analogue gives rise to the question of whether the distributed link services approach developed here can be mapped to distributed *knowledge* services. A step towards this would be to adopt an RDF model for linking where semantics of link captured in more specific manner than the existing link model.

Linkbase Fragment Mobility was another aspect of the identified system requirements that has not been fully explored. This is in part due to the observation that fragment and resolver mobility is not actually that useful when the affected participants in a query are local to each other, and in part due to the model where linkbase identity also implies resolver identity; the two entities tightly coupled.

However, should the participation extend to distribution wider than local link, the additional latency would be a major issue. In the current framework prototype that supports such a mechanism (section 6.2.3), a remote participant is not be able to request migration of the resolver/linkbase entity. Rather, the model employed is that the process owner pushes the process at some remote target explicitly.

An approach that achieves a similar result would be for the remote user to use a Linkbase Fragmenting utility to query a remote linkbase for all desired links at once and then capture the result set locally, perhaps publishing to a local resolver's linkbase. This alternative would work for static links (although would not be necessary in the referral-based LDAP prototype with query referral), however it would not cater with dynamic resolvers where links are computed as a result of a query, rather than looked up and matched against a linkbase.

When developing the prototype systems, the utility of generic linking when mated with the approach of matching non-local destination anchors to links in the Versioning and Representation dynamic linkbases was not expected to be quite so effective as was realised. When individual link resolvers published their resolved links to the shared space (e.g. in the tuplespaces models), processes that maintained the mapping of remote resources to local instances could be triggered to resolve remote destination links and offer local alternatives back to the space. This provided a mechanism of automatic link composition, rather than the query dispatcher or user explicitly having to ask for additional resolutions on non-local anchors.

After summarising the architecture recommendation for link services in pervasive computing environments, the sections that follow briefly discuss common observations beyond reflecting on the particular models.

7.1.1 Architecture Recommendation

The experiments enacted in the investigation of appropriate component decoupling, inter-component communication and computational models gives rise to two key characteristics for the enabling infrastructure of hypermedia link services in the pervasive computing environments that we have considered.

It is recommended that the architecture employ communication and cross-component computation that is decoupled in both space and time.

That is, the most appropriate technologies within the context of the assumptions made about the environment and the scenarios defined as of interest are a content-based routing communications infrastructure employing notification and subscription combined with a type-free data co-ordination language for the conveyance of hypermedia link data, link resolution queries and results, and other auxiliary data.

7.1.2 Perils of result (link) caching

When processes monitor links that are the result of Link Resolvers having processed a query on behalf of a user, the context conditions that resulted in the link being offered are unlikely to persist, or at least, not be captured. As a result, when capturing or caching a resolved link, the semantics or reasons for its resolution are lost.

This may not be an issue in that the reasons behind the link being a valid resolution result may be irrelevant to the user. The philosophy adopted by this work has been to develop the framework and link service infrastructure that *enables* link resolution and delivery pertaining local and spontaneously available resources and linkbases irrespective of the validity of the links after initial resolution and delivery.

Resolution caching and capture promote serendipity after the fact in that related resources may later be navigated to that otherwise would not have been discovered, although the fidelity of the association may be questionable.

Further consideration shall be to consider mechanisms by which the context that resulted in a particular link's resolution occurring being captured and returned as part of the Link Resolver result set, e.g. as an extension to the link model to include additional semantic descriptors as to the link's raison d'être.

7.1.3 Shared understanding

An associated issue to that above is that of shared understanding. Specifically, when providing the ability to partition the information space into, for

example, different linkbase themes or roles, there has emerged a need for processes, whether human or software in nature, to know what purpose each individual partition serves.

The approach taken with the framework developed in this thesis has been to enable the users of the system to utilise linkbase descriptor data, effectively providing a manual mechanism for partitioning the space. Should a user not require hyperstructure pertaining Topic A, then they would not enable linkbases whose descriptions suggest that they contain links regarding Topic A.

Further consideration of the aspects of link and linkbase metadata shall consider the role of the Resource Description Framework (RDF), specifically the Web Ontology Language (OWL) (McGuinness, 2002) as tools to provide a level of mechanisation to the adaptive interaction with resources as a result of computable meaning of their content in relation to one another.

7.1.4 Record and Replay

With event-based infrastructures, the interactions between components can be captured and, for example in the case of automation such as with timely repeat notification of the availability of a resource, replayed.

Whilst providing a messaging and triggering mechanism, this approach also facilitates an abstraction as to the nature of the source of the events. For instance, an event may be triggered by a user being sensed as present in a conference room, which could in turn result in the various conference services performing some arrival functionality, e.g. registering the person as a physical anchor in the case of a suitable utility (section 5.2.6.2).

The event has then blurred the boundary between physical and virtual participants of the system. An event that registers the presence of someone in the conference auditorium could also be generated by processes monitoring 'virtual visitors' to the on-line conference resources, as discussed in section 7.2.2.

7.2 Reflecting on the Scenarios

The observation that local resources, spontaneously available, have been shown to be discoverable, navigable and enriched with hyperstructure suggests that the adopted approach to the research and its outcomes have been a success.

This section considers the role of the scenarios beyond motivating and scoping the research undertaken.

7.2.1 Meeting Room

Throughout the development of the prototype link services, a method of testing has been to simulate or enact the scenarios, with regards section 4.4. When developing the framework and considering the interactions to support, the scenario that suited as a primary focus was that of the Meeting Room (section 4.2), given the nature of the information space therein and its suitability as a immediately available platform for testing.

The issues surrounding the incorporation of physical artefacts as anchors within the scenarios arose from a number of coincident research projects. As collaborators on the IST FEEL project (Jonnson, 2001), we developed a meeting room scenario in which the research interest was the management and coordination of notifications so as to manage their intrusiveness. Part of that research included a notion of context determination for the meeting at hand, and the use of RFID-tagged documents detected as being in use by virtue of being on the table. A natural extension of that process was to incorporate those physical documents as hyperstructure anchors, which led to the physical component of the Meeting Room scenario in section 4.2.

More recently, the local Signage activity has introduced the notion of people as physical anchors in a space, using technologies such as MIFARE contact-less smart cards and iButton docks for presence detection; and the deployment of screens to promote virtual annotation of posters and demonstrations at a conference (schraefel, 2004), which can then be consumed by a pDLS system.

Each of these systems that incorporate physical artefacts of some form require bespoke PhysicalTransmogrifier utilities to manage the resolution of physical anchor to some digital form. Further work is required that considers mechanisms that enable physical markers as link anchor targets as opposed the current realisation which is to serve dynamic digital content 'describing' the nature of the physical resource.

Example suitable technologies include Smart-Its (Homlquist, 2001) or Glow-pads due to MIME[10], both IST Disappearing Computer projects. The idea being that 'tagged' objects can be such that they make people aware of their location by, for example, emitting light or buzzing, when a link to them is resolved and traversed by the user. In the case of a Smart-Its realisation, it would be the role of the artefact's companion PhysicalTransmogrifier utility to cause a Smart-Its host controller to emit radio beacon that instructs the tag on the artefact to buzz.

7.2.2 Conference

The Conference scenario transpired to be natural extension of the Meeting Room, with a greater number of participants and, due to the nature of the information space, a greater number of instances of the same resource available. Whilst no evaluative enaction of this scenario was performed, simulation using an order of magnitude higher number of participants with similar resource bases demonstrated a good test of the use of Version & Representation linkbases for local instance discovery. A limitation of the current system has been identified as a result in that, when quite so many alternate versions of the same resource are available in the same space, the user interface becomes overloaded with alternative options. Further work is required on this interface aspect, calling for a filter mechanism in the GUI such that result overload does not occur.

The next natural extension, then, would be to distributed meetings, e.g. video conferences on the e-Science Access Grid as in the CoAKTinG project

---

[10] Multiple Intimate Media Environments project:
http://www.mimeproject.org/

(Buckingham Shum, 2002). Such an extension would require that the assumptions regarding the underlying network be re-addressed. For instance, it is unlikely that all participants will be in the same broadcast domain, and certainly not link-local as has been assumed by this work. Cursory analysis suggests that DNS-based service discovery for components could still be utilised, and technologies such as site-local addressing such as that offered by overlay networks could offer a workable solution, mimicking the conditions enjoyed by link-local networks, but with significantly greater communications latencies. Also, physical anchors and their participation in links would have an altogether different semantics, and possibly not be as meaningful or useful as they have been here.

7.2.3 Corridor

Finally, the enactment of the Corridor scenario has not been attempted with real users, its purpose more a motivating thought experiment than deployment target. However, it has provoked further discussion following the development of the MUD prototype and the use of physical context anchors such as barcodes and iButtons to tie artefacts or people to locations.

A shift in the topological model of the artefacts in the MUD away from being conceptual (where the 'rooms' are participant nodes and Linkbase Resolvers are Non-Player Characters, NPCs), and more towards a spatial representation of the physical space has highlighted a number of issues, and provoked the transfer of technology from the activities of this thesis to other research projects.

In the spatial MUD model, artefacts in the real world have associated digital representations, much more like the traditional use of a MUD, e.g. for gaming purposes. For example, a sign on a wall has a digital representation, perhaps including the same information digitally encoded. The MUD then models only those interactions that are of interest to the application at hand, captured by some sensing technology capable of detecting (and affecting) physical interactions, such as those discussed earlier in relation to the FEEL activity.

For example, as a user is sensed as being outside the lift, their NPC representation is transported to the room that models the area outside of the virtual lift in the MUD. Should there be a 'virtual' artefact in that space, for example a GeoNotes-like annotation (Espinoza, 2001), and an appropriate device in the vicinity (e.g. the user's hand-held, or a nearby wall-sign, both modelled in the MUD), then the virtual interaction that is the delivery of the annotation's contents to the appropriate modelled device could be reflected in the physical world with the result that the user becomes affected by a virtual interaction.

This type of cross-boundary linking and orchestrated interaction has formed the basis of the infrastructure realising the Equator Ambient Wood user trials. The Ambient Wood infrastructure employed the Elvin content-based notification framework as a sensor/actuator coupling interface between the physical environment, and a MUD virtual environment that maintained a digital model of artefacts and interactions of interest (Weal, 2003).

Scripted links in the MUD were triggered by the assertion of notifications pertaining sensed events leading to information being displayed or played in the wood, either in the form of the audio heard through hidden speakers, or images and voice-over information on the children's devices. The interactions were modelled as contextual hypermedia links, associating artefacts, locations, information and states (Thompson, 2003). This use of a MUD as a Hypermedia-based orchestration tool, interacting with the physical world through a coupling of distributed processes with the Elvin content-based messaging system, is a direct result of the findings of this thesis.

Bibliography

Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433, 1997. ISSN 1022-0038.

Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, 2000.

Robert M. Akscyn, Donald L. McCracken, and Elise A. Yoder. KMS: a distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–35, 1988.

Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper. RFC3171, BCP51: IANA Guidelines for IPv4 Multicast Address Assignments. IETF Request for Comments Document, Standards Track, August 2001. Available at *ftp://ftp.isi.edu/in-notes/rfc3171.txt*.

James Allan. Automatic hypertext link typing. In *Proceedings of the the seventh ACM conference on Hypertext*, pages 42–52. ACM Press, 1996. ISBN 0-89791-778-2.

Yair Amir, Claudiu Danilov, and Cristina Nita-Rotaru. High performance, robust, secure and transparent overlay network service. In *Proceedings of FuDiCo 2002: International Workshop on Future Directions in Distributed Computing*, June 2002.

Kenneth M. Anderson. A critique of the open hypermedia protocol. In *Proceedings of the Third Workshop on Open Hypermedia Systems. ACM Hypertext '97 Conference, Southampton, UK*, pages 11–7. Danish National Centre for IT Research, 80000 Århus C, Denmark, 1997a. Tech. Report SR-97-01.

Kenneth M. Anderson. Integrating open hypermedia systems with the World Wide Web. In *Proceedings of the eighth ACM conference on Hypertext*, pages 157–166. ACM Press, 1997b. ISBN 0-89791-866-5.

Kenneth M. Anderson. Client-side services for open hypermedia - getting past the "foo". In *Proceedings of the Fourth Workshop on Open Hypermedia Systems. ACM Hypertext '98 Conference, Pittsburgh, PA*, pages 15–21. Department of

Computer Science, 6700 Aalborg University, Esbjerg, Denmark, 1998. Tech. Report CS-98-01.

Kenneth M. Anderson, Richard N. Taylor, and Jr. E. James Whitehead. Chimera: Hypertext for heterogeneous software environments. In *Proceedings of the 1994 ACM European conference on Hypermedia technology*, pages 94–107. ACM Press, 1994. ISBN 0-89791-640-9.

K. Andrews, F. Kappe, and H. Maurer. Hyper-G and Harmony: Towards the next generation of networked information technology. In *Proceedings of the Conference companion on Human factors in computing systems*, pages 33–34. ACM Press, May 1995.

W.S. Ark and T. Selker. A look at human interaction with pervasive computers. *IBM Systems Journal*, 38(4):504–7, 1999.

H. Ashman, A. Garrido, and H. Oinas-Kukkonen. Hand-made and computed links, precomputed and dynamic links. In *Hypermedia - Information Retrieval - Multimedia '97 (HIM'97)*, pages 191–208, 1997.

IEEE Standards Association. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band. Technical report, IEEE, June 2003.

R. Bagrodia, W.W. Chu, L. Kleinrock, and G. Popek. Vision, issues, and architecture for nomadic computing. *IEEE Personal Communications*, pages 14–27, December 1995.

G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proceedings of the nineteenth IEEE International Conference on Distributed Computing Systems (ICDCS'99)*, pages 262–272, May 1999.

R. Barrett and P.P. Magilo. Intermediaries: An approach to manipulating information streams. *IBM Systems Journal*, 38(4):629–41, 1999.

Richard Bartle. Interactive multi-user computer games. Technical report, BT Martlesham Research Laboratories, December 1990. Annotated version available from *http://www.mud.co.uk/richard/imucg.htm*.

John Barton and Tim Kindberg. The challenges and opportunities of integrating the physical world and networked systems. Technical Report HPL-2001-18, Hewlett Packard Laboratories, 2001.

T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollerman. World-wide web: The information universe. *Computer Networks and ISDN Systems*, pages 454–9, 1992.

T. Berners-Lee, R. Fielding, and H. Frystyk. RFC1945: Hypertext Transfer Protocol – HTTP/1.0. IETF Request for Comments Document, Standards Track, May 1996. Available at *ftp://ftp.isi.edu/in-notes/rfc1945.txt*.

T. Berners-Lee, R. Fielding, and L. Masinter. RFC2396: Uniform Resource Identifiers (URI): Generic Syntax. IETF Request for Comments Document, Standards Track, August 1998. Available at *ftp://ftp.isi.edu/in-notes/rfc2396.txt*.

Tim Berners-Lee. Information management: A proposal. Technical report, CERN, March 1989. Internal Proposal, available at *http://www.w3.org/History/1989/proposal.html*.

Tim Berners-Lee. The World Wide Web - past, present and future. *Journal of Digital Information*, 1(1), 1997. Available at *http://jodi.ecs.soton.ac.uk/Articles/v01/i01/BernersLee/*.

Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Communications of the ACM*, 37(8):76–82, 1994. ISSN 0001-0782.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

T. Bickmore, A. Girgensohn, and J.W. Sullivan. Web page filtering and re-authoring for mobile users. *The Computer Journal*, 42(6), 1999.

Joel Birnbaum. Pervasive information systems. *Communications of the ACM*, 40 (2):40–41, 1997. ISSN 0001-0782.

G. Boriello and R. Want. Embedded computation meets the World Wide Web. *Communications of the ACM*, pages 59–66, May 2000.

S. Brandt and A. Kristensen. Web push as an internet notification service. In *Proceedings of the W3C workshop on Push Technology*, August 1997.

Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (xml). W3C Recommendation, December 1997. Available at *http://www.w3.org/TR/PR-xml-971208*.

Dan Brickley and Libby Miller. Foaf vocabulary specification. RDFWeb Namespace Document under Creative Commons License, August 2003. Available at *http://xmlns.com/foaf/0.1/*.

P. Brusilovsky. Methods and techniques of adaptive hypermedia. In *Proceedings of User Modeling and User Adapted Interaction*, volume 6, pages 87–129, 1996.

S. Buckingham Shum, D. De Roure, M. Eisenstadt, N. Shadbolt, and A. Tate. CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid. In *Proceedings of the Second Workshop on Advanced Collaborative Environments, Eleventh IEEE Int. Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland*, July 2002.

V. Bush. As we may think. *The Atlantic Monthly*, pages 101–8, July 1945.

O. Buyukkokten, H.C. Molina, and A. Paepcke. Focused web searching with PDAs. In *Proceedings of the Ninth International World-Wide Web Conference*, May 2000.

M. Bylund. sView - personal service interaction. Ph.lic. thesis, Computing Science Department, Uppsala University, Sweden, 2001.

B. Carmeli, B. Cohen, and A.J. Wecker. Personal information everywhere PIE. In *HT'00 - Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia Systems*, pages 252–3. ACM, May 2000.

L.A. Carr, D.C. De Roure, H.C. Davis, and W. Hall. Implementing an Open Link Service for the World Wide Web. *World Wide Web Journal*, 1(2), 1998.

Leslie A. Carr. *Structure in Text and Hypertext*. PhD thesis, University of Southampton, 1995.

Leslie A. Carr, Hugh C. Davis, David C. De Roure, Wendy Hall, and Gary J. Hill. Open information services. *Computer Networks and ISDN Systems*, 28(7/11): 1027–36, 1996.

Leslie A. Carr, David C. De Roure, Wendy Hall, and Gary J. Hill. The Distributed Link Service: A tool for publishers, authors and readers. *World Wide Web Journal*, 1(1):647–56, 1995.

N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32 (4):444–58, April 1989.

Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.

Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. IETF Internet Draft document, June 03a. Available at *http://www.ietf.org/internet-drafts/draft-cheshire-dnsext-dns-sd-01.txt*.

Stuart Cheshire and Marc Krochmal. Performing DNS queries via IP Multicast. IETF Internet Draft document, June 03b. Available at *http://www.ietf.org/internet-drafts/draft-cheshire-dnsext-multicastdns-02.txt*.

Stuard Cheshire (Ed.). Zero Configuration Networking (Zeroconf) Working group Charter. IETF Working Group Document, 2003. Available at *http://zeroconf.org/*.

James Clark (Ed.). XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999. Available at *http://www.w3.org/TR/1999/REC-xslt-19991116*.

Jeff Conklin. Hypertext: an introduction and survey. *Computer*, 20(9):17–41, 1987. ISSN 0018-9162.

Ward Cunningham. The Wiki Wiki Web: Welcome visitors. *Portland Pattern Repository*, October 2003. Available as *http://c2.com/cgi/wiki?WelcomeVisitors*.

N. Davies, K. Chevest, K. Mitchell, and A. Friday. Caches in the Air: Disseminating tourist information in the guide system. In *Proceedings of the Second IEEE*

Workshop on Mobile Computer Systems and Applications (WMCSA'99), February 1999.

H.C. Davis, A. Lewis, and A. Rizk. OHP: A draft proposal for a standard open hypermedia protocol. In *Proceedings of the Second Workshop on Open Hypermedia Systems. UCI-ICS Technical Report 96-10*, pages 27–53. University of California, Irvine, 1996.

Hugh Davis, Wendy Hall, Ian Heath, Gary Hill, and Rob Wilkins. Towards an integrated information environment with open hypermedia systems. In *Proceedings of the ACM conference on Hypertext*, pages 181–190. ACM Press, 1992. ISBN 0-89791-547-X.

Hugh C. Davis. *Data Integrity Problems in an Open Hypermedia Link Service*. PhD thesis, University of Southampton, 1995.

Hugh C. Davis. Referential integrity of links in open hypermedia systems. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space.structure in hypermedia systems*, pages 207–216. ACM Press, 1998. ISBN 0-89791-972-6.

Hugh C. Davis, Simon Knight, and Wendy Hall. Light hypermedia link services: a study of third party application integration. In *Proceedings of the 1994 ACM European conference on Hypermedia technology*, pages 41–50. ACM Press, 1994. ISBN 0-89791-640-9.

F. de Saussure. *Cours de Linguistique Générale*. Editions Payot, 1922. Due to *http://www.ccms-infobase.com/*.

Paul De Bra, Peter Brusilovsky, and Geert-Jan Houben. Adaptive hypermedia: from systems to framework. *ACM Computing Surveys (CSUR)*, 31(4es):12, 1999. ISSN 0360-0300.

Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems (TOCS)*, 8 (2):85–110, 1990. ISSN 0734-2071.

S. J. DeRose. Expanding the notion of links. In *Proceedings of the second annual ACM conference on Hypertext*, pages 249–257. ACM Press, 1989. ISBN 0-89791-339-6.

Steve DeRose, Eve Maler, and David Orchard. Xml linking language (XLink) version 1.0. W3C Recommendation, June 2001. Available at *http://www.w3.org/TR/xlink*.

D. C. De Roure and S. G. Blackburn. Content-based navigation of music using melodic pitch contours. *Multimedia Systems*, 8(3):190–200, 2000.

David C. De Roure, Leslie A. Carr, Wendy Hall, and Gary J. Hill. A distributed hypermedia link service. In *Proceedings of the Third International Workshop on*

Services in Distributed and Networked Environments, pages 156–61. IEEE, June 1996.

David C. De Roure, Samhaa El-Beltagy, Nicholas M. Gibbins, Leslie A. Carr, and Wendy Hall. Integrating link resolution services using query routing. In *Proceedings of the fifth Workshop on Open Hypermedia Systems (OHS5), ACM Hypertext'99 Conference*, pages 17–22, February 1999.

David C. De Roure, Nigel Walker, and Leslie A. Carr. Investigating link service architectures. In *HT'00 - Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia Systems*, pages 67–76. ACM, May 2000.

D.C. De Roure, W. Hall, S. Reich, A. Pikrakis, G.J. Hill, and M. Stairmand. An open framework for collborative distributed information management. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, volume 30 of *Computer Networks and ISDN Systems*, pages 624–5, Brisbane, Australia, April 1998.

P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider. RFC1835: Architecture of the WHOIS++ service. IETF Request for Comments Document, Standards Track, August 1995. Available at *ftp://ftp.isi.edu/in-notes/rfc1835.txt*.

A. Dey and G. Abowd. CyberMinder: A context-aware system for supporting reminders. In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000*, pages 172–86. Springer Verlag, September 2000.

Laura De Young. Linking considered harmful. In *Hypertext: Concepts, Systems and Applications, Proceedings of the Hypertext '90 Conference*, pages 238–49, 1990.

T. Dierks and C. Allen. RFC2246: The TLS Protocol Version 1.0. IETF Request for Comments Document, Standards Track, January 1999. Available at *ftp://ftp.isi.edu/in-notes/rfc2246.txt*.

A. Dillon, J. Richardson, and C. McKnight. The effect of display size and text splitting on reading lengthy text from the screen. *Behaviour and Information Technology*, 9(3):215–27, 1990.

K. Egevang and P. Francis. RFC1631: The IP Network Address Translator (NAT). IETF Request for Comments Document, Standards Track, May 1994. Available at *ftp://ftp.isi.edu/in-notes/rfc1631.txt*.

Samhaa R. El-Beltagy, Wendy Hall, David De Roure, and Leslie Carr. Linking in context. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 151–160. ACM Press, 2001. ISBN 1-59113-420-7.

Scott Elrod, Gene Hall, Rick Costanza, Michael Dixon, and Jim des Rivieres. Responsive office environments. *Communications of the ACM*, 36(7):84–5, July 1993.

D. C. Engelbart. A conceptual framework for the augmentation of man's intellect. *Vistas of Information Handling*, 1, 1963.

Douglas C. Engelbart, Richard W. Watson, and James C. Norton. The augmented knowledge workshop. In *AFIPS Conference Proceedings*, volume 42, pages 9–21. National Computer Conference, June 1973.

F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund. GeoNotes: Social and navigational aspects of location-based information systems. In Abowd, Brumitt, and Shafer, editors, *Proceedings of Ubicomp 2001: Ubiquitous Copmuting, International Conference, Atlanta, Georgia*, pages 2–17. Springer-Verlag, September 2001.

D. Estrin, R. Govindan, and J. Heidemann. Introduction to embedding the internet. *Communications of the ACM*, pages 38–41, May 2000.

Geraldine Fitzpatrick, Tim Mansfield, Simon Kaplan, David Arnold, Ted Phelps, and Bill Segall. Instrumenting the workaday world with elvin. In *Proceedings of ECSCW'99*, pages 431–451, Copenhagen, Denmark, September 1999. Kluwer Academic Publishers.

A Fountain, W Hall, I Heath, and H Davis. Microcosm: An open model for hypermedia with dynamic linking. In A Rizk, N Streitz, and J Andre, editors, *Hypertext: Concepts, Systems and Applications, Proceedings of ECHT'90, Paris, November 1990*, pages 298–311. Cambridge University Press, 1990.

A. Fox and E.A. Brewer. Reducing WWW latency and bandwidth requirements by real-time distillation. In *Proceedings of the Fifth International World-Wide Web Conference*, Paris, France, May 1996.

D. Frohlich, E. Tallyn, N. Linketscher, B. Signer, N. Adams, and P. Luff. Paper++ Delivery 8, Initial Paper++ Assessment. Technical report from the IST DC Paper++ Project, April 2002.

V. Fuller, T. Li, J. Yu, and K. Varadhan. RFC1519: Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy. IETF Request for Comments Document, Standards Track, September 1993. Available at *ftp://ftp.isi.edu/in-notes/rfc1519.txt*.

D. Gelernter. Generative communication in Linda. *TOPLAS*, 7(1):80–112, 1985.

Carole Goble and Leslie Carr. COHSE: Informed WWW link navigation using ontologies. In *Proceedings of the IEEE Colloquium on "Lost in the Web"*, November 1999.

Stuart Goose, Jonathan Dale, Wendy Hall, and David De Roure. Microcosm TNG: a distributed architecture to support reflexive hypermedia applications. In *Proceedings of the eighth ACM conference on Hypertext*, pages 226–227. ACM Press, 1997. ISBN 0-89791-866-5.

Kaj Grønbæk, Niels Olof Bouvin, and Lennert Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the eighth ACM conference on Hypertext*, pages 146–156. ACM Press, 1997. ISBN 0-89791-866-5.

Kaj Grønbæk, Jens A. Hem, Ole L. Madsen, and Lennert Sloth. Designing Dexter-based cooperative hypermedia systems. In *Proceedings of the fifth ACM conference on Hypertext*, pages 25–38. ACM Press, 1993. ISBN 0-89791-624-7.

Kaj Grønbæk, Jannie F. Kristensen, Peter Ørbæk, and Mette Agger Eriksen. 'Physical hypermedia': organising collections of mixed physical and digital material. In *Proceedings of the Fourteenth ACM conference on Hypertext and Hypermedia*, pages 10–19. ACM Press, 2003. ISBN 1-58113-704-4.

Kaj Grønbæk and Randall H. Trigg. Design issues for a Dexter-based hypermedia system. In *Proceedings of the ACM conference on Hypertext*, pages 191–200. ACM Press, 1992. ISBN 0-89791-547-X.

Kaj Grønbæk and Randall H. Trigg. Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):40–49, 1994. ISSN 0001-0782.

Kaj Grønbæk and Randall H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of Hypertext '96, The Seventh ACM Conference on Hypertext*, pages 149–60. ACM, March 1996. ISBN 0-89791-778-2.

Kaj Grønbæk, Peter Posselt Vestergaard, and Peter Ørbæk. Towards geo-spatial hypermedia: Concepts and prototype implementation. In *Proceedings of the thirteenth conference on Hypertext and hypermedia*, pages 117–126. ACM Press, 2002. ISBN 1-58113-477-0.

E. Guttman, C. Perkins, J. Viezades, and M. Day. RFC2608: Service Location Protocol (SLP), version 2. IETF Request for Comments Document, Standards Track, June 1999. Available at *ftp://ftp.isi.edu/in-notes/rfc2608.txt*.

Bernard J. Haan, Paul Kahn, Victor A. Riley, James H. Coombs, and Norman K. Meyrowitz. IRIS hypermedia services. *Communications of the ACM*, 35(1):36–51, 1992.

F. Halasz and M. Schwartz. The Dexter hypertext reference model. In *Proceedings of the Hypertext Standardisation Workshop*, pages 95–133, Gaithersburg, MD, January 1990. US Government Printing Office.

Frank Halasz. Seven issues: Revisited. In *Proceedings of the third annual ACM conference on Hypertext*, pages 171–171. ACM Press, 1991. ISBN 0-89791-461-9.

Frank G. Halasz. Reflections on Notecards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7):836–52, 1988.

Wendy Hall. Ending the tyranny of the button. *IEEE Multimedia*, 1(1):60–8, 1994.

Wendy Hall, Mark J. Weal, Ian Heath, Gary B. Wills, and Richard M. Crowder. Flexible interfaces in the industrial environment. In *Proceedings of International*

Conference Managing Enterprises-Stakeholders, Engineering, Logistics and Achievement (ME-SELA'97), pages 453–460, July 1997.

Uwe Hansmann, Lothar Merk, Martin S. Nicklous, and Thomas Stober. *Pervasive Computing Handbook*. Springer Verlag, 2001.

Linda Hardman, D.C.A. Bulterman, and G. van Rossum. The Amsterdam Hypermedia Model: Adding time and context to the Dexter model. *Communications of the ACM*, 37(2):50–62, February 1994.

Ian Heath. *An Open Model for Hypermedia: Abstracting Links from Documents*. PhD thesis, University of Southampton, UK, 1992.

G. Hill and W. Hall. Extending the Microcosm model to a distributed environment. In *Proceedings of the ACM European conference on Hypermedia technology (ECHT '94)*, pages 32–40. ACM Press, September 1994.

Steve Hitchcock, Freddie Quek, Leslie Carr, Wendy Hall, Andrew Witbrock, and Ian Tarr. Towards universal linking for electronic journals. *Serials Review*, 24 (1):21–33, 1998.

P. Hoffman, L. Masinter, and J. Zawinski. RFC2368: The mailto URL scheme. IETF Request for Comments Document, Standards Track, July 1998. Available at *ftp://ftp.isi.edu/in-notes/rfc2368.txt*.

L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Proceedings of UBICOMP 2001*, September 2001.

IBM. MQSeries Everyplace for Multiplatforms: Introduction. Technical Document number GC34-5843-05, September 2001. Available at *http://www-3.ibm.com/software/integration/mqfamily/library/*.

Internet Assigned Numbers Authority. Internet Multicast Addresses (Under continual review), September 2001. Available at *http://www.iana.org/assignments/multicast-addresses*.

Internet Mail Consortium. vCard - The Electronic Business Card Version 2.1. IMC Technical note, September 1996. Available at *http://www.imc.org/pdi/vcard-21.txt*.

J. Ioannidis, D. Duchamp, and G.Q. Maguire Jr. IP-based protocols for mobile interworking. In *Proceedings SIGCOMM'91*, pages 235–45. ACM, September 1991.

C.G Jonsson, J. Matsson, P. Werle, F. Kilander, M. Ciobano, and P. Lonnquist. The design of the physical aspect of an interactive space (I-Space) with particular emphasis on the enabling of non-intrusive behaviours. Deliverable from the IST DC FEEL Project, March 2001.

R. José and N. Davies. Scalable and flexible location-based services for ubiquitous information access. In H.-W. Gellersen, editor, *Proceedings of Handheld and*

Ubiquitous Computing. First International Symposium, HUC'99, number 1707 in Lecture Notes in Computer Science, pages 52–66. Springer Verlag, Karlsruhe, Germany, September 1999.

Charles J. Kacmar and John J. Leggett. Proxhy: a process-oriented extensible hypertext architecture. *ACM Transactions on Information Systems (TOIS)*, 9(4): 399–419, 1991. ISSN 1046-8188.

Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1), 2003.

S.A. Karim and P. Hovell. Everything over IP – an overview of the strategic challenge in voice and data networks. *BT Technology Journal*, 17(2), April 1999.

J. Kempf. Building the 4th generation cellular networks. Presentation, Sun Microsystems, Berkeley Multimedia, UCB, April 2000.

T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive*, 1(1):70–81, March 2002.

Tim Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. In *Proceedings of the eleventh international conference on World Wide Web*, pages 191–199. ACM Press, 2002. ISBN 1-58113-449-5.

Tim Kindberg and John Barton. A Web-based nomadic computing system. *Computer Networks*, 35(4):443–456, 2001.

Tim Kindberg, Rakhi Rajani, Mirjana Spasojevic, and Ella Tallyn. Pulp Computing (Demonstration). In *Proceedings of The Fifth International Conference on Ubiquitous Computing, UbiComp03*, October 2003.

Simon J. Knight. *Abstracting Anchors from Documents*. PhD thesis, University of Southampton, UK, 1996.

M. Kolon and W.J. Goralski. *IP Telephony*. McGraw Hill, September 1999.

J. Leggett and R. Killough. Issues in hypertext interchange. *Hypermedia*, 3(3): 159–86, 1991.

Paul H. Lewis, Hugh C. Davis, S. Griffiths, Wendy Hall, and Robert J. Wilkins. Media-based navigation with generic links. In *Proceedings of the Seventh ACM Conference on Hypertext*, pages 215–223. ACM Press, March 1996.

Paul H. Lewis, Hugh C. Davis, Steven R. Griffiths, Wendy Hall, and Robert J. Wilkins. Content based retrieval and navigation with images in the microcosm model. In *Proceedings of the International Conference on Multimedia Communications*, pages 86–90. The Society for Computer Simulation International, 1995. ISBN 156555-046-3.

P. Loshin. *IPv6 Clearly Explained*. Morgan Kaufmann, 1999.

D. Lowe and W. Hall. *Hypermedia and the Web*. John Wiley and Sons, 1999.

Kathryn C. Malcolm, Steven E. Poltrock, and Douglas Schuler. Industrial strength hypermedia: requirements for a large engineering enterprise. In *Proceedings of the third annual ACM conference on Hypertext*, pages 13–24. ACM Press, 1991. ISBN 0-89791-461-9.

Friedemann Mattern and Mahmoud Naghshineh, editors. *Pervasive Computing, First International Conference, Pervasive 2002*, volume 2414 of *Lecture Notes in Computer Science*, August 2002. Springer.

Hermann Maurer. *Hyper-G now Hyperwave: The Next Generation Web Solution.* Longman Group United Kingdom, 1996.

Donald L. McCracken and Robert M. Akscyn. Experiences with the ZOG human computer interface system. *Journal of Man-Machine Studies*, 21:293–310, 1984.

Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. W3C Candidate Recommendation, August 2003. Available at *http://www.w3.org/TR/2003/CR-owl-features-20030818/*.

N. Meyrowitz. The missing link: why we're all doing hypertext wrong. *The society of text: hypertext, hypermedia, and the social construction of information*, pages 107–114, 1989.

Danius T. Michaelides, David E. Millard, Mark J. Weal, and David C. De Roure. Auld Leaky: A contextual open hypermedia link server. In *Proceedings of the Seventh Workshop on Open Hypermedia Systems, ACM Hypertext 2001 Conference*, 2001.

David E. Millard, Luc A.V. Moreau, Hugh C. Davis, and Siegfried Reich. Standardizing hypertext: Where next for OHP? In Siegfried Reich and Kenneth M. Anderson, editors, *Proceedings of the Sixth Workshop on Open Hypermedia Systems and the Second Workshop on Structural Computing, San Anotio, Texas, USA*, number 1903 in Lecture Notes in Computer Science. Springer Verlag, August 2000a.

D.E. Millard, L.A.V. Moreau, H.C. Davis, and S. Reich. FOHM: A fundamental open hypertext model for investigating interoperability between hypertext domains. In *HT'00 - Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia Systems*, pages 93–102. ACM, May 2000b.

R. Moats. RFC2141: URN syntax. IETF Request for Comments Document, Standards Track, May 1997. Available at *ftp://ftp.isi.edu/in-notes/rfc2141.txt*.

G. Moore. VLSI: Some fundamental challenges. *IEEE Spectrum*, 16:30, 1979.

A. Murphy, G. Picco, and G.-C. Roman. Lime: A middleware for physical and logical mobility. In *Proceedings of the twenty-first International Conference on Distributed Computing Systems (ICDCS-21)*, pages 524–536, May 2001.

T.H. Nelson. *Literary Machines.* published by the author, 1981.

Theodor H. Nelson. *Computer Lib/Dream Machines*. published by the author, Swarthmore, PA, 1976.

Theodor H. Nelson. *Literary Machines*. Sausalito Press, 87.1 edition, 1987.

Theodor H. Nelson. Managing immense storage. *Byte Magazine*, 13(1):225–38, January 1988.

Theodore H. Nelson. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Proceedings of the Twentieth National Conference*, pages 84–100, 1965.

Peter J. Nürnberg and Helen Ashman. What was the question? reconciling open hypermedia and World Wide Web research. In *Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*, pages 83–90. ACM Press, 1999. ISBN 1-58113-064-3.

Peter J. Nürnberg, John J. Leggett, and Uffe K. Wiil. An agenda for open hypermedia research. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space.structure in hypermedia systems*, pages 198–206. ACM Press, 1998. ISBN 0-89791-972-6.

J. Oikarinen and D. Reed. RFC1459: Internet Relay Chat protocol. IETF Request for Comments Document, Standards Track, May 1993. Available at *ftp://ftp.isi.edu/in-notes/rfc1459.txt*.

Reinhard Oppermann, Marcus Spect, and Igor Jaceniak. Hippie: A nomadic information system. In H.W. Gellersen, editor, *Proceedings of Handheld and Ubiquitous Computing, HUC99*, number 1707 in Lecture Notes in Computer Science, pages 330–4. Springer Verlag, 1999.

Kasper Østerbye and Uffe Kock Wiil. The Flag taxonomy of open hypermedia systems. In *Proceedings of the the seventh ACM conference on Hypertext*, pages 129–139. ACM Press, 1996. ISBN 0-89791-778-2.

Kevin R. Page, Don Cruickshank, and Roure De Roure. Its about time: link streams as continuous metadata. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 93–102. ACM Press, 2001. ISBN 1-59113-420-7.

D. Papp. Dealing with imperfect information in poker. Masters Thesis of the Department of Computing Science, University of Alberta, 1998.

C. Partridge, T. Mendez, and W. Milliken. RFC1546: Host anycasting service. IETF Request for Comments Document, Standards Track, November 1993. Available at *ftp://ftp.isi.edu/in-notes/rfc1546.txt*.

Gian Pietro Picco, Amy L. Murphy, and Gruia-Catalin Roman. LIME: Linda meets mobility. In *Proceedings of the International Conference on Software Engineering*, pages 368–377, May 1999.

Morgan N. Price, Gene Golovchinsky, and Bill N. Schilit. Linking by inking: trailblazing in a paper-like hypertext. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space.structure in hypermedia systems*, pages 30–39. ACM Press, 1998. ISBN 0-89791-972-6.

Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. RFC1918: Address Allocation for Private Internets. IETF Request for Comments Document, Standards Track, February 1996. Available at *ftp://ftp.isi.edu/in-notes/rfc1918.txt*.

Antoine Rizk and Louis Sauter. Multicard: an open hypermedia system. In *Proceedings of the ACM conference on Hypertext*, pages 4–10. ACM Press, 1992. ISBN 0-89791-547-X.

David S. Rosenblum and Alexander L. Wolf. A design framework for internet-scale event observation and notification. In M. Jazayeri and H. Schauer, editors, *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)*, pages 344–360. Springer–Verlag, 1997.

M. Satyanarayanan. A catlayst for mobile and ubiquitous computing. *IEEE Pervasive*, 1(1):2–5, March 2002.

R. Scheifler. Jini connection technology. In *Proceedings of Pervasive Computing 2000: New IT Industry Conference*, Gaithersburg, MD, January 2000.

Albrecht Schmidt. Implicit human computer interaction through context. *Personal Technologies*, 4(2), June 2000.

Jean Scholtz. Ubiquitous computing goes mobile. *Mobile Computing and Communications Review*, 5(3):32–8, July 2001.

m.c. schraefel *et al.* Connecting Physical+Temporal Events to Digital Contexts. Submitted to CHI2004, September 2003.

H. Schulzrinne. Personal mobility for multimedia services in the Internet. In *Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Services*, Berlin, Germany, March 1996.

B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of Queensland AUUG Summer Technical Conference*, pages 243–255, September 1997. Brisbane, Australia. Available as *http://www.dstc.edu.au/Elvin/doc/papers/auug97/AUUG97.html*.

A Selvin, S Buckingham Shum, M Sierhuis, J Conklin, B Zimmermann, C Palus, W Drath, D Horth, J Domingue, E Motta, and G Li. Compendium: Making meetings into knowledge events. In *Proceedings of Knowledge Technologies 2001*, 2001.

Patrick A. S. Sinclair, Kirk Martinez, David E. Millard, and Mark J. Weal. Links in the palm of your hand: Tangible hypermedia using augmented reality. In

Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, pages 127–136, 2002.

Jennifer Smith. *The MUD FAQ*. The MUD Connector, 1990. Continually revised, available at *http://www.mudconnect.com/mudfaq/*.

J.R. Smith, R. Mohan, and C. Li. Scalable multimedia delivery for pervasive computing. In *Proceedings of ACM Multimedia 99*, pages 131–40, Orlando, October 1999.

Ted Smith and Steve Bernhardt. Expectations and experiences with hypercard: a pilot study. In *Proceedings of the sixth international conference on Systems documentation proceedings*, pages 47–56. ACM Press, 1988. ISBN 0-89791-336-1.

P. David Stotts and Richard Furuta. Petri-net-based hypertext: document structure with browsing semantics. *ACM Transactions on Information Systems (TOIS)*, 7(1):3–29, 1989. ISSN 1046-8188.

Sun Microsystems. JavaSpaces: White Paper. Technology Briefing, 1998.

C.P. Thacker, E.M. McCreight, B.W. Lampson, R.F. Sproull, and D. Boggs. Alto: A personal computer. In D.P. Siework, C.G. Bell, and A. Newell, editors, *Computer Structures: Principles and Examples*. McGraw-Hill, New York, USA, 2nd edition, 1981.

The Salutation Consortium. Salutation Architecture Specification V2.1. Technical Briefing, 1998. Available from *http://www.salutation.org/*.

Mark K. Thompson, Mark J. Weal, Danius T. Michaelides, Don G. Cruickshank, and David C. De Roure. MUD Slinging: Virtual Orchestration of Physical Interactions. Technical Report ECSTR-IAM03-001, Department of Electronics and Computer Science, University of Southampton, January 2003.

R. Tolksdorf. Laura: A coordination language for open distributed systems. In *Proceedings of the thirteenth IEEE International Conference on Distributed Computing Systems*, pages 234–239, 1993.

M. Tzagarakis, N. Karousos, D. Chritodoulakis, and S. Reich. Naming as a Fundamental Concept of Open Hypermedia Systems. In *HT'00 - Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia Systems*, pages 103–112. ACM, May 2000.

A. Vaha-Sipila. RFC2806: URLs for Telephone Calls. IETF Request for Comments Document, Standards Track, April 2000. Available at *ftp://ftp.isi.edu/in-notes/rfc2806.txt*.

Andries van Dam. Hypertext '87: Keynote address. *Communications of the ACM*, 31(7):887–895, 1988. ISSN 0001-0782.

K. van Laerhoven, N. Villar, A. Schmidt, H.-W. Gellersen, M. Håkansson, and L. E. Holmquist. Pin&play: The surface as network medium. *IEEE Communications Magazine*, 41(4):90–96, April 2003.

Louis Visser (Ed.). Data elements and interchange formats – information interchange – representation of dates and times. International Organization for Standardization Standard, December 2000.

R. Want, T. Pering, G. Borriello, and K.I. Farkas. Disappearing hardware. *IEEE Pervasive*, 1(1):36–47, March 2002.

Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI*, pages 370–377, 1999.

Roy Want, Bill Schilit, Norman Adams, Rich Gold, Karin Petersen, John Ellis, David Goldberg, and Mark Weiser. The PARCTAB ubiquitous computing experiment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, March 1995.

S. Waterhouse. JXTA Search: Distributed Search for Distributed Networks. White Paper, May 2001. Available from *http://search.jxta.org/*.

Mark J. Weal, Danius T. Michaelides, Mark K. Thompson, and David C. De Roure. The Ambient Wood Journals - Replaying the Experience. In *Proceedings of Hypertext'03, The fourteenth conference on Hypertext and Hypermedia*, pages 20–27, 2003.

Mark J. Weal, David E. Millard, Danius T. Michaelides, and David C. De Roure. Building narrative structures using context based linking. In *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*, pages 37–38, 2001.

Stuart Weibel, Erik Jul, and Keith Shafer. PURLs: Persistent uniform resource locators. Technical report, Online Computer Library Center (OCLC), 1999. Available as *http://purl.oclc.org/OCLC/PURL/SUMMARY*.

M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1991.

M. Weiser. Some computer science issues in ubiquitous computing. *Communicaitons of the ACM*, 36(7):74–83, July 1993.

M. Weiser, R. Gold, and J.S. Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM Systems Journal*, 38(4):693–7, 1999.

J. Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9, 1966.

Jim Whitehead, Paul De Bra, Kaj Grønbæk, Deena Larsen, John Leggett, and monica m. c. schraefel. Seven issues, revisited. In *Proceedings of the thirteenth conference on Hypertext and hypermedia*, pages 171–171. ACM Press, 2002. ISBN 1-58113-477-0.

E. James Whitehead, Jr. and Yaron Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the web. In *Proceedings of the Sixth European*

Conference on Computer Supported Cooperative Work (ECSCW'99), pages 291–310, September 1999.

Uffe Kock Wiil and Kasper Østerbye, editors. *Proceedings of the ECHT'94 Workshop on Open Hypermedia Systems*, September 1994. Department of Computer Science, Aalborg University, Denmark. Tech. Report R-94-2038.

Uffe Kock Wiil and Kasper Østerbye. Using the flag taxonomy to study hypermedia system interoperabilty. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space.structure in hypermedia systems*, pages 188–197. ACM Press, 1998. ISBN 0-89791-972-6.

Robert J. Wilkins. *The Advisor Agent: a Model for the Dynamic Integration of Navigation Information within an Open Hypermedia System.* PhD thesis, University of Southampton, UK, 1994.

A. Williams. Requirements for Automatic Configuration of IP Hosts. IETF Internet Draft document, September 02. Available at *http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-reqts-12.txt*.

P. Wyckoff, S.W. McLaughry, T.J. Lehman, and Ford D.A. TSpaces. *IBM Systems Journal*, 37(3), August 1998.

J. Yang and I. Kriaras. Wireless VoIP: Opportunities and challenges. In Hong Va Leong, Wang-Chien Lee, Bo Li, and Li Yin, editors, *Proceedings of Mobile Data Access. First International Conference, MDA'99*, number 1748 in Lecture Notes in Computer Science, pages 3–13. Springer Verlag, Hong Kong, China, December 1999.

N. Yankelovich, Bernard J. Haan, Norman K. Meyrowitz, and S.M. Drucker. Intermedia: The concept and the construction of a seamless information environment. *IEEE Computer*, 21(1):81–96, 1988.

W. Yeong, T. Howes, and S. Kille. RFC1777: Lightweight Directory Access Protocol. IETF Request for Comments Document, Standards Track, March 1995. Available at *ftp://ftp.isi.edu/in-notes/rfc1777.txt*.