UNIVERSITY OF SOUTHAMPTON

Continuous Metadata Flows for Distributed Multimedia

by

Kevin R. Page

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering, Science, and Mathematics

School of Electronics and Computer Science

April 2011

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

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

Doctor of Philosophy

by Kevin R. Page

The practical use of temporal multimedia has increased markedly in recent years as enabling technologies for the distribution and streaming of media have become available. As a part of this trend, hypermedia systems and models have adapted accordingly to incorporate such distributed multimedia for presentation.
Structured interpretation of information has long been a fundamental feature of both open hypermedia systems and knowledge systems. Metadata, in its many forms, has become the cornerstone for providing this structured knowledge above and beyond basic data and information.
This thesis presents the rationale and requirements for continuous metadata, which supports the metadata accompanying distributed multimedia throughout the lifecycle of streamed media, from generation, through distribution, to presentation. Throughout this process it is the temporal and continuous nature of the metadata which is paramount. A conceptual framework for continuous metadata is proposed to encapsulate these principles and ideas.
Continuous metadata and the associated framework enable the development, in particular, of real-time, collaborative, semantically enriched distributed multimedia applications. Experience building one such system using continuous metadata is evaluated within the framework. An ontology is developed for the system to enable the collation, distribution, and presentation of structure aiding navigation of multimedia, and it is shown how continuous metadata utilising the ontology can be distributed using multicast.

Contents

List of Figures

DECLARATION OF AUTHORSHIP

I, Kevin R. Page, declare that the thesis entitled "Continuous Metadata Flows for Distributed Multimedia", and the work presented in it are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed; where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself:
  - the work described in chapter 5 was undertaken as part of the CoAKTinG project[1], in which Compendium and BuddySpace were developed at the Open University, I-X Process Panels at the University of Edinburgh, and the Meeting Replay tool by a team at the University of Southampton including myself. The CoAKTinG ontology and distributed replay control are specifically my own contributions, and the focus of discussion in chapter 5;

[1] http://www.aktors.org/coakting/

- parts of this work have been published as:

  Page, K. R., Cruickshank, D., and De Roure, D. (2001) *Its About Time: Link Streams as Continuous Metadata*. In Proceedings of The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01), pp. 93-102.

  Beales, R., Cruickshank, D., De Roure, D., Gibbins, N., Juby, B., Michaelides, D. T. and Page, K. R. (2001) *The Pipeline of Enrichment: Supporting Link Creation for Continuous Media*, in Openness, Structural Awareness, and Adaptivity: International Workshops, pp. 47-58.

  Page, K., Juby, B., Beales, R. and De Roure, D. (2001) *Continuous Metadata*. In Proceedings of the 2nd Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting, pp. 264-9.

  De Roure, D. C., Cruickshank, D. G., Michaelides, D. T., Page, K. R. and Weal, M. J. (2002) *On Hyperstructure and Musical Structure*. In Proceedings of The Thirteenth ACM Conference on Hypertext and Hypermedia (Hypertext 2002), pp. 95-104.

  Bachler, M. S., Buckingham Shum, S. J., De Roure, D. C., Michaelides, D. T. and Page, K. R. (2003) *Ontological Mediation of Meeting Structure: Argumentation, Annotation, and Navigation*. In Proceedings of 1st International Workshop on Hypermedia and the Semantic Web (HTSW2003)

  Bachler, M., Buckingham Shum, S., Chen-Burger, J., Dalton, J., De Roure, D., Eisenstadt, M., Komzak, J., Michaelides, D., Page, K., Potter, S., Shadbolt, N. and Tate, A. (2004) *Collaboration in the Semantic Grid: a Basis for e-Learning*. In Proceedings of Grid Learning Services workshop (GLS 2004) at the 7th International Conference on Intelligent Tutoring Systems (ITS 2004), pp. 1-12.

Page, K. R., Michaelides, D. T., Buckingham Shum, S. J., Chen-Burger, Y., Dalton, J., De Roure, D. C., Eisenstadt, M., Potter, S., Shadbolt, N. R., Tate, A., Bachler, M., and Kozmak, J. (2005) *Collaboration in the Semantic Grid: a Basis for e-Learning.* In Journal of Applied Artificial Intelligence 19(9-10), pp. 881-904

Signed:

Date:

Acknowledgements

For support in my work, thanks to members of the Pervasive Computing and Networks theme past and present, and to my collaborators in CoAKTinG, particularly Danius Michaelides. For support at home, thanks to my parents, family, and friends. Finally, for his supervision, my thanks to David De Roure, who gave me the freedom, time, and encouragement to find and explore the areas of research that continue to interest and motivate me.

Chapter 1

Introduction

Hypertext systems have added dimensions such as navigation and annotation to electronically stored documents, dramatically increasing the worth and flexibility of the information held within. As these distributed systems have developed, they have grown to encompass a much richer multimedia environment, embedding temporal content such as audio and video, and often streaming the media to the user for real-time viewing.

Hypermedia, and more generally multimedia, systems have also developed in regards to the way they deal with knowledge. The use of metadata is one tool by which we can impose ordered structure upon data and information, raising interesting parallels between knowledge systems and hypermedia with regards to more generic structural computing. The use of metadata is, however, confined to generation and presentation within distributed multimedia applications.

This thesis presents the case and motivation for *continuous metadata*, which enables the development of distributed hypermedia applications that can support temporal media as successfully as non-streamed content, by extending the reach of structure, through metadata, into the distribution stage of multimedia applications.

The ideas and techniques introduced are particularly valuable when supporting live, collaborative, and semantically rich distributed applications. The

development of one such system, as part of the CoAKTinG project, is described and provides an evaluation for the framework, and a validation of continuous metadata. An OWL ontology is created, and forms the basis of an implementation of continuous metadata using Semantic Web technologies. Finally, a proof-of-concept tool extends the use of continuous metadata onto multicast networks.

1.1 Outline of Thesis

Chapter 2 gives an overview of the area of research, bringing together the elements needed to present continuous media to the user. It begins with background information about hypertext systems, including Open Hypermedia Systems and the World Wide Web, giving details of their development and introducing more recent research areas such as the semantic web. The requirements for multimedia applications are presented, with particular attention paid to the needs of temporal multimedia in a distributed environment, including Quality of Service, multicast, and real-time protocols. We investigate how continuous content is presented within hypermedia systems, from the Amsterdam Hypermedia Model to the SMIL markup language, and how it can be annotated through the use of metadata. Metadata itself is examined, and an overview of the knowledge frameworks within which it is used, finally linking back to the structure of hypermedia systems and the use they are put to, in combination, in the semantic web.

Chapter 3 builds upon these ideas, presenting the novel notion of continuous metadata. By analysing a 'lifecycle' perspective of streamed media within distributed multimedia applications, it is shown how metadata to augment such multimedia is inadequately supported during the distribution stage of the cycle. Continuous metadata is introduced to allow the use of metadata throughout the generation, distribution, and presentation stages. Metadata is shown to be a generalisation of structure, and that the concepts supporting structure in Open

Hypermedia are applicable to continuous metadata.

Chapter 4 expands upon the idea of continuous metadata: supporting information used in combination with the streamed multimedia and transported in a correspondingly timely manner. While chapter 2 details the existing support for the delivery and presentation of temporal multimedia data, chapter 4 introduces a conceptual framework for the distributed processing and delivery of associated continuous metadata. This framework has the ability to bring together metadata from multiple sources both at the users' viewing platform and at intermediate processing nodes, allowing the creation of powerful real-time filter chains. After the initial discussion of point-to-point connections between framework elements, a multicast scenario is presented.

Chapter 5 recalls experiences from the CoAKTinG project, in which wide-ranging trials were conducted incorporating the ideas supporting continuous metadata. The project validates the generalisation from link streams to continuous metadata in its use of Semantic Web tools and technologies - specifically the use of ontologies - to create a navigational hypertext for the detail-rich video recordings of collaboration. The tools and techniques used in this scenario are evaluated in the context of the conceptual framework, where a significant deficiency is found to be the lack of support for multicast; a proof-of-concept demonstrator is implemented to overcome this shortfall in chapter 6.

1.2 Contributions

The key contributions of this thesis can be summarised as follows:

- An analysis of metadata for supporting distributed multimedia: through the introduction of a 'lifecycle' model, an appraisal of the key elements required to support streaming multimedia is given; a comparative review is undertaken with regard to metadata, and continuous metadata is presented as a resolution for the weaknesses which are identified. Continuous

metadata is also shown to be a generalisation of link streams to semantic structure (chapter 3).

- The creation of a conceptual framework to evaluate continuous metadata and define its requirements (chapter 4), and validation of this framework and continuous metadata through experience: an OWL ontology is developed as the lynchpin for enabling continuous metadata in a Semantic Grid collaboration system and, following trials, the system is evaluated in terms of the framework (chapter 5); a proof-of-concept tool is shown to validate the use of continuous metadata over multicast (chapter 6).

The work in this thesis has contributed in part or full to the following publications:

Page, K. R., Cruickshank, D., and De Roure, D. (2001) *Its About Time: Link Streams as Continuous Metadata*. In Proceedings of The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01), pp. 93-102.

Beales, R., Cruickshank, D., De Roure, D., Gibbins, N., Juby, B., Michaelides, D. T. and Page, K. R. (2001) *The Pipeline of Enrichment: Supporting Link Creation for Continuous Media*, in Openness, Structural Awareness, and Adaptivity: International Workshops, pp. 47-58.

Page, K., Juby, B., Beales, R. and De Roure, D. (2001) *Continuous Metadata*. In Proceedings of the 2nd Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting, pp. 264-9.

De Roure, D. C., Cruickshank, D. G., Michaelides, D. T., Page, K. R. and Weal, M. J. (2002) *On Hyperstructure and Musical Structure*. In Proceedings of The Thirteenth ACM Conference on Hypertext and Hypermedia (Hypertext 2002), pp. 95-104.

Bachler, M. S., Buckingham Shum, S. J., De Roure, D. C., Michaelides, D. T. and Page, K. R. (2003) *Ontological Mediation of Meeting Structure:*

Argumentation, Annotation, and Navigation. In Proceedings of 1st International Workshop on Hypermedia and the Semantic Web (HTSW2003)

Page, K. R., Michaelides, D. T., Buckingham Shum, S. J., Chen-Burger, Y., Dalton, J., De Roure, D. C., Eisenstadt, M., Potter, S., Shadbolt, N. R., Tate, A., Bachler, M., and Kozmak, J. (2005) *Collaboration in the Semantic Grid: a Basis for e-Learning.* In Journal of Applied Artificial Intelligence 19(9-10), pp. 881-904

Chapter 2

Hypertext, Multimedia, and Metadata

This chapter provides a survey of work in the areas related to the author's research. In the first section an overview of hypertext and hypermedia is given, including Open Hypermedia Systems and the World Wide Web. The requirements and solutions for distributing real-time multimedia across a network are presented, followed by a section on integrating such temporal data within a hypermedia environment. Finally, it is shown how metadata can be used to augment this multimedia material, from basic markup to the structuring information and knowledge representation.

These topics, each a distinctive field of research in its own right, form a boundary around the content of this thesis. Two principle themes run through this chapter, the first of which are the requirements for supporting distributed, streamed, multimedia both in the network and for presentation. The second is that of structuring data to provide information, from hypertexts and hypermedia, through annotation using metadata, to the ontologies of the Semantic Web.

The intersection of these themes forms the motivation for *continuous metadata* in chapter 3.

2.1 Hypertext

2.1.1 The Origins of Hypertext

The concept of a hypertext system was first described in Vannevar Bush's seminal 1945 paper 'As We May Think' [39]. A scientific adviser to Roosevelt's wartime administration, Bush had overseen many of the rapid advances in technology driven by the fight for military superiority. With the end of conflict in sight, Bush hoped that this expansion of learning would continue apace with more peaceful purpose, but realised that no one man would be able to access the ever increasing body of knowledge unaided. He foresaw the emergence of an information society, and hypothesised on the construction of a mechanical device he called the 'Memex', which would allow its user to record, recall and share both existing archives and newly authored material. Furthermore, the Memex provided navigation of this data in an associative manner, mimicking the human memory through an "intricate web of trails". In this way, Bush had realised the advantage of using machines not just for computation, but for organising and structuring information as knowledge.

While Bush set the agenda for what would become hypertext research, a Memex-like system would not be realised for a further two decades until the work of Engelbart [67]. At the same time Nelson [105] was independently developing ideas to create a global network of inter and intra-connected structured documents, explicitly conceived as a real incarnation of the Memex. And it was during development of this 'Xanadu' system that the term 'hypertext' was first conceived to describe the non-linear text used in this environment.

2.1.2 The Development of Hypertext: Structure through Linking

As the availability of computing power grew through the 1970s to the present day, so too did the development of hypertext systems. In his eminent 1987

survey, Conklin identified the advantages he saw in hypertext systems [45]:

Ease of tracing references System support for tracing links means that it is equally easy to follow a reference forwards or backwards.

Ease of creating new references Without changing the original document, users can build their own network of references or annotate someone else's.

Information structuring Otherwise unstructured information can have both hierarchical and non-hierarchical organisations imposed overlaid onto it; multiple organisations can structure the same information in different ways.

Global views Structural overviews (such as table of content style views) can be provided over large amounts of data, which can be mixed with use of local (detailed node or page) views, allowing easier restructuring of complex documents.

Customised documents Segments of text can be threaded together in many ways to use the same document for multiple functions.

Modularity of information The same segment of text can be referenced from several places as it becomes useful, without the need to duplicate or overlap the content within it.

Consistency of information References are attached to a segment of text, and as such are transferred with the text should it be moved, maintaining consistency.

Task stacking The user can have several paths of enquiry active and displayed at the same time. Any given path can be unwound to the original task.

Collaboration When several authors work together on a document, their comments and annotations are tightly interwoven between each other, and with the document.

Conklin surmised that that links were the essential feature of a hypertext system, providing the *structural core* to them, and that other common components only built upon this structure or facilitated its maintenance.

He also highlighted two major problems inherent in hypertext systems:

Disorientation When navigating a non-linear hypertext, there is a real danger the user will become "lost in hyperspace" while trying to navigate the extra degrees of freedom hypertext offers. Nielsen found that over half the readers of a document in the early hypertext system HyperCard became confused by "where they were" (the context-in-the-large problem) [106].

This user disorientation can manifest itself in three ways[66]:

1. Users not knowing where to go next.
2. Users knowing where to go but not knowing how to go there.
3. Users not knowing where they are within the overall structure.

The interface of a hypertext system must present navigation aids to the user to minimise the disorientation.

Cognitive Overhead It may be difficult for a user to overcome the additional mental burden needed to create, name, and keep track of links when authoring or browsing a hypertext system; the system should no distract the user from the material within. Similarly, if the interface only offers a window onto a small fragment of the document, the user can have difficult keeping track of the proceeding and following segments (context-in-the small).

It is interesting to note that both these weaknesses are directly related to be the ability of a system to managed, or at least present a manageable interface, to the structure held by the hypertext, not problems intrinsic to the fundamental structure.

2.1.3 Separation of Links and Open Hypermedia

While the importance of overlaid structure and links was clear, the method by which a particular system should hold and manipulate the hyperstructure in relation to the underlying document was not. This poses the question: where should the links be stored? There are three solutions to this problem [51]:

1. Embed all the linking information in the referencing content document itself, within the source anchor (where an anchor is the point in the document a link is to or from).
2. Embed a persistent marker within the content document, but retain the linking information externally.
3. Store all linking information, including source and destination anchors, externally, with no added mark-up of the content document.

Open Hypermedia Systems, or OHSs, follow the third behaviour, which allows them to link into and out of read-only material. It also enables the use of alternative, or multiple, views through different link databases (linkbases), which can be distributed and independently maintained, and criteria based selection of linkbases (e.g. using a linkbase according to the user's context).

The separation of linking information forms one of the five defining features of an OHS [52]:

1. The system does not impose any markup upon the data.
2. The system can integrate with third party tools.
3. The system can be distributed across networks and hardware platforms.
4. The system does not distinguish between readers and authors.
5. The system should allow the easy addition of extra functionality.

A further perspective on these features is in how they deal with the hypertext (linking) structure in and between documents; in this way OHSs advocate the separation of explicit structure from the document, and prescribe that this structure should be distributable in querying, authoring, and maintenance.

2.1.4 Scaling Hypertext Systems

For large scale hypertext systems such as those envisioned by Bush and Nelson scalability of the system must be present in both the data and application domains [95].

The inherent abstraction of system components in OHSs can allow distribution with relatively little modification [87] thus addressing some application scalability issues, though often on a relatively local (LAN) level. Other work enabled larger scale distribution of link servers through the use of document caching [93], but in doing so lost the consistency of an OHS.

Data scalability requires the use of multiple linkbases, each storing large quantities of data. While research has been undertaken to build systems on an industrial scale (tens of thousands of links) [9], scaling and distribution continues to be a problem area for OHSs.

2.1.5 The Dexter Hypertext Reference Model

By the early 1990s numerous hypertext systems existed but failed to communicate effectively amongst each other. The Dexter Hypertext Reference Model [81] was developed to rationalise and make explicit the concepts presented by these systems in one model which could then be used for standard comparison and interaction.

The Dexter model represents a hypertext system as three layers:

1. the **within-component** layer stores details of the content and structure of a particular node or document.

2. the **storage** layer stores the hypertext structure (links and nodes).

3. the **runtime** layer stores the information needed so that the user can view and interact with the hypertext.

Between the within-component and storage layer lies the **Anchoring** interface, which defines a mechanism for addressing locations within an individual component. Similarly, between the storage and runtime layers the **Presentation Specifications** interface defines how the runtime layer should represent the objects in the storage layer. The model also introduces three components:

1. **Atomic components** contain a presentation specification, a semantic description of the component, an anchor and its content type. An **Anchor** is composed of a unique identifier for reference, and a data-dependent anchor value which specifies a part of the atomic component.

2. **Composite components** allow a collection of other components (atomic, composite or link) to be represented as a single component.

3. A **Link** is a connection among *two or more* components. It contains a list of **Specifiers**, each of which represents an end-point of the link using an anchor, a direction and a presentation specification.

At the time of its inception, the Dexter model took concepts from many hypertext systems, such that no one system implemented them all. Since then, systems have been built to implement the model [77] and this has highlighted some weaknesses in it, such as lack of support for embedded or dynamic ("generic") links. In particular the model is lacking when representing large-scale distributed hypertext systems, since all applications comprising the within-component layer must be known to the system and model their data with the Dexter storage layer.

2.1.6 Structural Computing

The evolution of hypertext systems can be generalised into five stages, where each stage abstracts "non-hypermedia essential" properties from the system, allowing this functionality to become "open" and/or distributed [109]:

Monolithic Systems In early systems, such as HyperCard, a single program provided all the functionality. The user interface, linking mechanism and interface to a basic file store were all provided by one program, allowing no explicit distribution.

Abstraction of Applications The development of "open" hypermedia systems began with systems where an open set of separate applications communicated, using a common linking protocol, with a centralised (but still monolithic) open link engine. Although only the user interface had been abstracted, this was still an important step which allowed these applications to become distributed [114].

Abstraction of Stores Hypermedia systems began including a separate layer between the link engine and the back-end filestore. These systems, such as Hyperbase[132], often utilised the functionality and flexibility of a database management system for the filestore. Having abstracted the storage layer away from the link engine this too could be recognised as a single, well-defined, program - the "link service". However, hyperbases did not normally result in an open or distributed storage layer; rather, structure was promoted to a first class entity within that storage layer.

Abstraction of Behaviours Open Hypermedia Systems (OHSs) as we know them today were developed when the link traversal behaviour was also abstracted and opened, resulting in systems such as Chimera [10] and Microcosm [52].

Abstraction of Link Services The final step is the abstraction of the link service itself into a generic structure server [108].

Following this progression through, the final abstraction suggests that the navigation of information spaces found in Open Hypermedia Systems is just one problem in the paradigm of structural computing and metainformatics. Links have always been a first class element in hypermedia; structural computing asserts the primacy of more general structure over data [109], and applies principles and analysis historically associated with hypertext to the management and processing of structure in broader terms, throughout computer science. In this domain, any service should be examined from the viewpoint of it providing a structural service (e.g. hypertext systems providing a structural navigation service).

Other work in the field supports the view that structure can be abstracted across different forms of hypertext. The Fundamental Open Hypermedia Model grew out of standardisation and inter-operation work in the Open Hypermedia Systems Working Group [100]; but instead of specific protocols for the three most important hypertext domains (navigational, spatial, and taxonomic), FOHM defines a generalised model for expressing structure which can be applied across and between all of the domains [101].

2.1.7 The World Wide Web

While lacking the advanced functionality of OHSs [69] the World Wide Web [19], or WWW, has enjoyed phenomenal popularity as a hypertext system and is used by millions around the globe. The Web might be considered little more than a glorified file retrieval service [107]; it is a data-centric system which manipulates flat files, compared with the OHS philosophy of treating links as first class objects and manipulating structure. WWW links are embedded within source documents, are singular and uni-directional, and the system lacks the capabilities for general user link authoring. This means that it is only possible to have one (embedded and permanently associated) linkbase for each page, making link consistency and integrity an order of magnitude harder to maintain.

It is, however, highly distributed, so that many millions of people throughout the world can access the same data. The use of a simple and standardised text-based file format [118], an easy to understand global naming system [20], and an easy to implement data protocol [68] means that all browsers on all platforms can access the same information. This allows inter-operability on a scale never realised with OHSs, where the system power grows directly in proportion to the number of users. Since the system itself is not structurally aware it can use more traditional and readily available technologies in the back-end, while the user can still hold and interpret *implied* structure as they are using the WWW.

Another trade-off in return for scalability is the lack of integrity in the WWW hyperstructure [11]. Because the linkbase is embedded within individual distributed documents, with no central management, it is extremely difficult to maintain the validity of links and ensure referential integrity. Web users are frequently confronted with links to documents that have moved or no longer exist (and now recognised in popular culture from the resulting HTTP error "404: Not Found"). However, this "scruffy works" approach can also be considered one of the reasons for the WWWs ability to scale.

In terms of explicit structure and linking support, the WWW is a poor relative of earlier, more traditional, hypertext systems. However, its simpler design has bypassed the scalability problems associated with OHSs, providing the foundation for the Web's success. Structure within documents is mostly relegated to presentational, rather that semantic, use; but that is not to say that the Web is devoid of structure, merely that its limited, relatively flat, structure has become powerful almost entirely due to scale.

2.1.8 Linking OHSs to the WWW

Much work has now been undertaken integrating the structural awareness and functional sophistication of OHSs with the distribution and ease of use of the WWW. The aim is to create a combined and superior solution which can use

simple standards to increase inter-operability while still allowing effective link consistency and management.

By considering the major architectural elements common to both systems (clients, server, protocols and data), solutions can be categorised as intersections of these elements [8]:

OHS to WWW Data translation tools, such as those produced for Microcosm, create linked HTML documents from an OHSs content. This essentially allows authoring to continue within the OHS with (less functional) publishing of the information on the Web.

Using a WWW Client as an OHS Client allows a Web browser to become an OHS enabled application so that HTML pages can be linked to other OHS applications. Microcosm achieved this using its "Universal Viewer", as did Chimera [10] with its wrapper system.

Using a WWW Server as an OHS Client allows a normal Web user to access the functionality of an OHS. Early versions of the Distributed Link Service (DLS) [41] allowed a user to select a region of text which would be passed to the OHS link resolvers. Any relevant links in the OHS were then returned as an HTML page by the DLS web server. Later less intrusive versions of the system used an HTTP proxy server to resolve links and add them to an HTML document as it was delivered to the WWW browser.

Hybrid Systems use a combination of the above techniques. The Devise HyperMedia (DHM) system [77] and its extensions to interact with the WWW [76] attempted to integrate clients into a web browser using Java applets, Javascript and browser plug-ins. These extensions would then use web server CGI-scripts to fetch link information from the DHM server and present it within the browser.

Chimera has also been extended in a hybrid manner [8], integrating the WWW into its OHS through use of standard Web protocols between

system components and providing Chimera functionality to web users through applets.

Running an OHS Server as a WWW Server presents a similar interface to the user as the mechanisms described in the previous section, but offers a greater degree of efficiency and flexibility. Despite the added development effort needed to add Web server functionality, systems such as Hyper-G [93] can accept requests from standard Web browsers and return information translated into HTML.

Amongst systems which augment the functionality of the WWW there is always a common goal to add, display and aid the visualisation of the structure the Web usually lacks. The Arakne Framework [29] is an attempt to provide a conceptual model and implementation within which the various strategies for Web augmentation can be unified. By splitting systems into three layers (content, service and structure) Arakne can be used to model existing augmentation strategies and, through study and analysis of common features, provide a basis for the infrastructure of future tools.

2.1.9 Development of the WWW

While Open Hypermedia Systems have continued to evolve, so too has the nature of the WWW, mostly under the auspices of the World Wide Web Consortium (W3C) [141]; the WWW is now by far the dominant medium for publicly accessible Internet-based resources. A number of developments are taking place which could allow the Web to absorb many of the more sophisticated features developed for hypermedia systems and beyond, incorporating ideas from the knowledge management and artificial intelligence (AI) communities. The first of these has been the adoption of a more rigorous and well defined syntax, with the associated ability to modularise, be self descriptive, and allow a greater separation of presentation and linking from content. The second phase of this development is the rise of the "Semantic Web", which is discussed in section 2.4.6.

The most important and fundamental change to the WWW, and indeed many other document based systems, is a move towards the use of the Extensible Markup Language (XML) [34]. XML is a simplified subset of SGML which defines a way to describe and mark up structured data, using tags similar to those popularised by HTML. Furthermore, it is a "meta-language" and can be used to describe further XML based languages. The power of XML lies in its simplicity and flexibility; XML forms the foundation of all the technologies being used to build the "web tomorrow" (figure 2.1). The use of XML Schema enables distributed definition of further XML based languages, which used in conjunction with the XML Namespace mechanism can scope the interpretation and meaning of a resource or document. There are now many, many, standardised XML dialects in use for structuring information and communications, and many more are created and used on an ad-hoc basis.

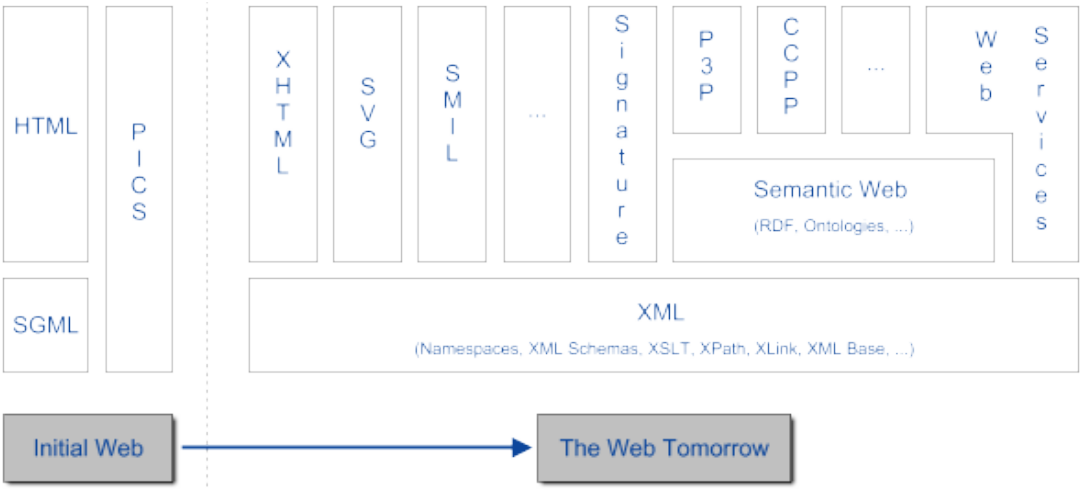

FIGURE 2.1: Development of the WWW
(from [141])

XML has been used to create a new version of the Hypertext Markup Language called XHTML [115], which is both well-formed and modular - it is broken down into different functional sections (structure, text, hypertext, tables, links and so on) which can be used and implemented only when needed. It has also freed HTML from becoming an all consuming "one-markup-fits-all" behemoth, as the needs for further functionality on the WWW can be provided independently, yet still be defined using XML and integrated with web documents while

maintaining separate namespace integrity.

This rationalisation of HTML has, in turn, allowed increased separation between the purely presentational aspects of a web document and the structure of the information that rendering is normally derived from. Cascading Style Sheets (CSS) [28] enable authors and users to flexibly attach styles (font, colour, alignment etc.) to elements of structure within a document or system, allowing the information (in XML) to be organised without pollution by display requirements. This also allows disciplined development of adaptive applications, where the same information might be presented differently depending on where and how it is used.

Other XML languages developed for Web presentation include the Scalable Vector Graphics (SVG) [31] specification to describe graphics, and a temporal presentation markup (SMIL, explored in more detail in section 2.3.4). In both cases the capability to render the XML to a user must be provided at the application level (and is often incorporated into web browsers).

As well as separation of presentation from content, three XML based mechanisms have been developed to enable the use of more powerful hypertextual linking and associations on the WWW, and the ability to separate links from content: XPath, XPointer and XLink.

The XML Path Language, XPath [44], is used to address parts of an XML document not by its syntax, but through a precise hierarchical tree representation of nodes and elements. The XML Pointer Language, XPointer [49], is needed to extend this functionality to allow addressing using defined document structure, pre-defined ID tags, arbitrary areas or a combination of methods for robustness. For instance, when a user selects an area of a document which crosses multiple parts of different XPath nodes, XPointer can express this zone in terms of its broader location definitions. Addressing of documents in this way is a requirement of providing links (and other markup) to a document without embedding the links themselves - a defining feature of earlier OHSs.

The XML Linking Language, XLink [58], allows navigational expressions to be placed within XML documents and can use XPointers to define its endpoints. In addition to the one-way in-line links of HTML, XLink supports bi-directional, multi-way, typed links, which can be stored separately to the documents they reference (in either simple files or linkbases). Link resolution can either be requested by the user or processed when the document is loaded, and the document retrieved via the link can either replace the initial document or be inserted within it.

In addition to the XML based standards for documents and linking, the HTTP protocol has been extended to allow Distributed Authoring and Versioning [92]. While WebDAV [74] is primarily used to aid publishing of Web documents, its developers believe it could evolve into a more generic distributed global filesystem.

The new functionality presented above should allow next generation Web tools to take on many of the features previously reserved for OHSs [82][27], and with XML and WebDAV set to be used in other, non-WWW, applications (such as office tool suites) there are exciting possibilities for providing true hypermedia functionality in these arenas too. The Semantic Web, through use of structured metadata and knowledge, takes Web development a stage further, and is described in section 2.4.6.

Finally, it is interesting to note that although the WWW is beginning to support mechanisms for the more advanced features of navigational hypermedia, the Open Hypermedia community is now working with other forms of structural hypermedia which are difficult, if not impossible, to describe in terms of links [78].

2.2 Multimedia

2.2.1 Real-time Distributed Multimedia

While textual hypermedia is a useful tool for conveying and navigating information, the use of multiple media types allows authors to communicate their ideas more effectively. Pictures and diagrams greatly enhance a document, and the use of temporal media such as audio and video create even more powerful presentations, especially when used in a hypermedia system.

The first multimedia systems were based on single machines, but research soon turned to designing distributed versions which, from an application viewpoint, can be broadly classified into three types [80]:

Presentational Applications present users with real-time multimedia transfered across the network from a storage or broadcast server to the user's display device, and include video-on-demand and news-on-demand.

Conversational Applications involve real-time multimedia communication between multiple users, such as video-phones. The real-time requirements are more stringent than for presentational applications since there is a greater sensitivity to delay.

Combined Applications are the most common type of multimedia system, and have both presentational and conversational aspects. In areas such as video-conferencing it is expected that users can both present pre-produced content and communicate interactively with other users.

From a more systems oriented perspective, four support resources can be identified which are required for such distributed multimedia applications [5]:

Explicit Support for Continuous Media - Multimedia in a system can be categorised as *static*, having no temporal dimension, or *continuous*, with an implied temporal dimension and real-time characteristics. Continuous

media, often known as multimedia *stream*s, requires support and resources from the underlying system for the duration of the media at the presentation rate required.

Quality Of Service - Continuous media uses large amounts of system resources for the duration of its presentation, so there must be a mechanism to specify and reserve the level of support needed.

Synchronisation - Multimedia applications require synchronisation to control event ordering and the timing of interactions between presentation elements.

Group Communication - Multimedia applications require the ability to present to and communicate with groups of collaborating users, rather than one user at a time.

For the first 20 years of its life the Internet was mainly used for email and file transfer, but now its infrastructure and protocols must adapt to cope with the additional demands of real-time multimedia [129]. It is currently not very efficient at carrying content such as audio and video [83], and fails to fulfil all of the requirements outlined above.

The rest of this section gives an overview of some of the solutions for providing real-time distributed multimedia over the Internet, as summarised by the protocol stacks shown in figure 2.2.

2.2.2 Quality of Service

Quality of Service, or QoS, can be expressed using four groups of parameters [42]:

Media Parameters represent the characteristics of the data being transferred. *Quantitative* characteristics include the type of media (e.g. live audio, or still video) and the media format (e.g. MPEG-1, MJPEG). *Qualitative*

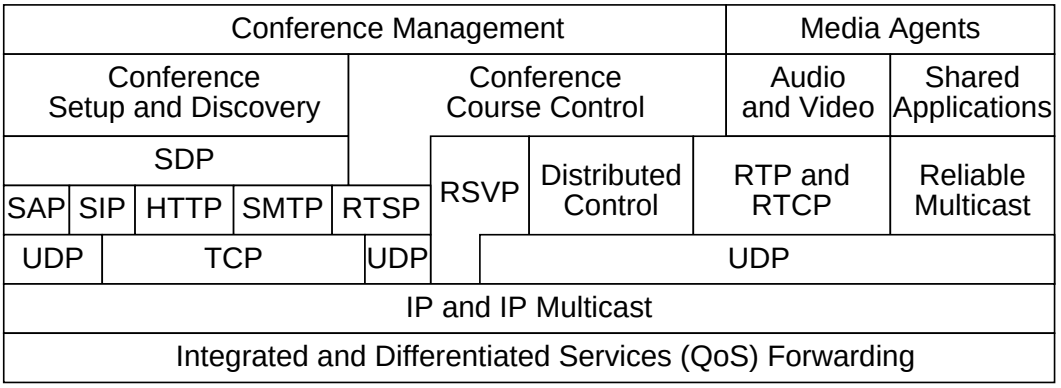

FIGURE 2.2: Internet multimedia protocol stacks
(from [83])

characteristics represent the quality of a particular media, such as resolution, speed and colour depth for video.

Delivery Parameters or real-time performance requirements, include start time, end time, packet delay and delay-jitter.

Value Parameters attach a cost to a delivery based upon the expense of reserving resources, transfer and delivery of data and the cost of producing the media. Such costs are determined on a per-session basis and likely to vary through time as supply and demand bandwidth levels are balanced. Attaching a value helps regulate over-subscription to services; if there were no cost there would be no rationale to choose anything but the highest level of service.

Fault-Tolerance Parameters define different classes of service commitment, each of which can deliver traffic with a specified reliability and timeliness.

While initial research provided QoS at specific layers of the network stack, there have since been many proposals for both specific and generic QoS architectures [12][80] which provide a complete framework for enabling QoS. When used for distributed multimedia, it is important that an architecture provides *end-to-end* guarantees, where it is the architecture which manages the QoS issues from application to application [12].

The Internet has traditionally provided *best-effort* delivery of data, in that there is no guarantee that any particular packet will arrive at a destination in order and in a timely manner, if at all. On a part of the network which is not congested, data packets will not become queued at routing nodes along the link, packets will not be lost, and delays will be minimal. But multimedia data is often large, and when transfered over the network leads to congestion resulting in packet loss and delay, the very things real-time traffic are vulnerable to.

For such distributed multimedia, a *better-than-best-effort* service is required [83] which must be supported at all nodes along the route across the Internet. While no such scheme has yet been deployed on an Internet-wide scale, there are two main QoS proposals which deal with the problem in two different manners:

Integrated Services - The Intserv architecture [32] is based on the idea of marking each packet from a particular flow of data from application to application with a *flow label*, which routers can use to identify it. Each routing element is QoS aware and can offer various levels of service above best-effort: controlled delay, predicted delay, and guaranteed delay. Within this framework a reservation protocol, RSVP [33][145], is used to set up resources across the network. For a particular application (marked by its flow label) RSVP attempts to traverse the network requesting the required level of service from each routing node, repeating the traversal with different requests until the destination is successfully reached. Each node is then honoured to reserve that level of service for the duration of that particular flow, and this requires routers to support complex state tables for each flow they handle. Due to this complexity, and the need for every node along the route to fully implement the architecture, Intserv and RSVP are not widely deployed across the Internet, although a flow label field exists in the header for the next generation Internet Protocol, IPv6 [54].

Differentiated Services - The Diffserv architecture [25] instead defines a relatively small number of *traffic classes*. Within the diffserv network routers then prioritise packets of higher classes to an agreed level of service,

so an application need merely request a particular traffic class. This leads to simpler routers which only examine the class of packet, rather than maintain knowledge of individual flows. Separate diffserv sub-networks can be managed with different internal QoS and billing policies, but at the junction with other networks packets can still easily be transferred by mapping to the traffic classes. Again, a traffic class field is present in the header for IPv6 [54].

While QoS architectures are essential in delivering distributed real-time multimedia, it is important to remember that QoS needs not only to be met in measurable objective parameters, but also in terms of a user's subjective requirements [80]. Users' needs must be translated into technical parameters [42] and the QoS measured in relation to a user's perception [72]. There is a complex relationship between perception and understanding; for example, a user may overlook a poor quality (e.g. through low frame rate) element of a multimedia presentation if the conceptual core of the presentation remains intact.

2.2.3 Multicast

Multicast routing [56] addresses the need distributed multimedia applications have for group communication. Multicasting is the delivery of data packets to a group of hosts, and while this is widely available and used across local networks, specialist routing is needed to traverse a collection of networks such as those that constitute the Internet. Multicasting multimedia data to a group of hosts provides two main benefits over unicasting [55]:

- The data will be delivered with greater efficiency since transmission of replicated information across common network routes is avoided. The architecture is scalable since information about members of the multicast group is kept at the routers nearest to the relevant members.
- Location-independent addressing can be used. The sending application transmits to a multicast group rather than a specific address, and the

receiving application registers to receive from a multicast group rather than a specific address. Senders and receivers do not need to know about each other or care about the network topology between them [83].

Internet multicast uses the connectionless (datagram) UDP [117] transport protocol to transmit data, so packet order and delivery are not guaranteed. However multicast is still appropriate for use with multimedia since the loss of small amounts of data in a video or audio stream can be overcome when a suitable higher level transport protocol is used [129].

Despite this, and the higher control overheads of a connection oriented protocol, there are several proposals for and implementations of reliable multicast protocols (e.g. [70],[113]) which guarantee the delivery of packets, although no single solution has been universally successful. The design of these protocols differs dramatically, with varying degrees of 'reliability' (normally with regard to ordering and causality rather than non-delivery).

Deployment of multicast is slowly expanding from the Multicast Backbone, or MBone, onto the Internet at large, particularly through next-generation projects such as Internet 2; moving from intra-domain connected with tunnels to large scale inter-domain native multicast routing [7]. With the uptake of IPv6 [54] multicast will be available as a standard provision of the Internet Protocol.

2.2.4 Transport Protocols

While multicast provides a means for group communication, the Internet needs still to provide recognition of real-time data paths [130]. Such transport protocols must provide explicit support for continuous media, work well in a multicast environment, allow for media synchronisation, and at the present time generally deal without true QoS guarantees. Several higher level transport protocols have been developed which in combination can successfully transfer media such as MPEG-2 video across the Internet [16].

A streaming media transport protocol (and applications that use it) must deal with three major requirements:

Tolerating Packet Loss The Internet, and the Internet Protocol (IP), operate on a best-effort basis; there is no guarantee that data packets will arrive, or arrive in order. The connection based Transmission Control Protocol (TCP) is commonly used above the IP (hence TCP/IP), and does provide such guarantees (e.g. for delivering web pages, HTTP uses TCP). However, it does so at a cost of increased processing overheads.

Each TCP packet sent by a transmitting node is identified with a sequence number, and after it is sent, also stored in a transmission buffer. If a packet successfully reaches the receiving node, that receiver sends an acknowledgement back to the transmitter, which can safely remove the packet with that sequence number. If the packet is lost, then the sender can take the packet from the transmission buffer and re-transmit.

The delay introduced by this re-transmission makes connection oriented protocols unsuitable for real-time media transport. By the time TCP detects a lost packet, requests its retransmission from the sender, and receives the packet, it is likely to have lost its moment in any real-time video (or that the video must pause, creating jitter). Instead, applications must make the best of the data that is received, and compensate at the application and media encoding layers (see section 2.2.5).

Controlling Delay Because IP is a packet based networking protocol, each router in the network deals with a varying number of packets, from various different sources, over subsequent periods of time, and each packet sent from a transmitter to a receiver may travel via a different combination of routers. So in addition to data loss, the period between receiving packets can vary and packets may not arrive in order, even if they were sent consecutively at a constant rate. If audio and video is sent at a constant rate, for correct playback it must be received at a constant rate; the undesired effect of playback with a differing rate is known as *jitter*.

This is most easily dealt with by a *playout buffer* [119], which delays the use of a packet after it has been received, effectively averaging out any jitter. While the associated *playout delay* for presentational applications is normally tolerable, conversational and combined applications must strike a fine balance between playout to compensate jitter, and limiting delay for interactivity.

Dynamic Throughput Adaption Most Internet congestion occurs on only a limited section of the path between the transmitter and receiver (a common bottleneck [26]), and due to its packet based nature, the level of this congestion can vary over time (dependant on other uses of that network segment). Quality of Service frameworks can compensate against this to a degree through packet prioritising, but even fully reserved QoS is susceptible to a renegotiation of service over the period of a long streaming media transmission.

So any transport protocol should provide some method of feedback by which the streaming application can adapt the rate of its transmission to compensate for bandwidth availability (normally as a function of perceived media quality).

RTP, the Real-time Transport Protocol [127], is a relatively simple protocol for transferring continuous media packets, and usually uses UDP as an underlying delivery mechanism. Use of a connectionless datagram transport protocol keeps packet latency down and makes RTP ideally suited to function over multicast, but does mean that there is no notion of a permanent connection or of reliable and timely delivery [129]; any compensation for packet loss must be met by higher layers in the network and application stack. Although mechanisms exist to cope with packet loss (section 2.2.5), for RTP to successfully carry a media flow at all there must be enough capacity in the network to accommodate the data, and this must be provided independently of RTP (possibly through a QoS architecture).

Every source of RTP data is identified by a source identifier, and every packet

produced by that source carries that marker in its header. If several RTP streams are combined and retransmitted, then the new amalgamated packets contain the source identifiers of all the contributing sources. RTP packet headers also contain a payload type field, and a timestamp field which is set at a rate dependent on the payload type. Other fields aid detection of lost and out of sequence packets.

The RTP Control Protocol [127], **RTCP**, is a companion protocol required by RTP that handles the delivery monitoring of data packets. All participants in an RTP session periodically send RTCP control packets to all other members of the session using the same transport mechanism as the RTP data itself. There are several types of these control packets, each with a different purpose [131]:

- Sender Reports (SR) are transmitted if a user is also sending an RTP media stream. It describes the amount of data sent, and timestamp to actual time correlation information (which is used for synchronisation between multiple streams).
- Receiver Reports (RR) are transmitted if a user is receiving an RTP media stream, and can give a stream source information about the current delay to the user.
- Source Descriptor (SDES) packets are used for simple session control, and contain a globally unique identifier used to list users in a session.

By monitoring the SR and RR control packets an RTP source receives feedback on the quality of data arriving at a receiver, allowing dynamic adjustment of transmission rate by applications higher in the stack.

RTCP also contains a mechanism to throttle back the number of control messages that are sent. If packets were simply sent at a set period, then as the number of session participants grew so to would the control traffic, until it overwhelmed the media traffic. To prevent this, the period between RTCP transmissions is increased with the number of users.

RTSP, the Real Time Streaming Protocol [128], is classified as an application level protocol, but is more of a framework with which to utilise RTP/RTCP. Using an HTTP-like syntax, RTSP provides a way to control a stream of data being transmitted from a multimedia server in both multicast and unicast environments. Although RTP is usually used as the media delivery mechanism, RTSP allows the use of other underlying transports while adding the ability to control and synchronise media streams with a common set of commands. By maintaining state about the various incoming media, RTSP allows a user to start, stop, and jump between various points in the stream, providing the final level of control over continuous media in this Internet framework.

2.2.5 Multimedia Content Encoding

With the use of an unreliable network transport protocol and (better-than-) best-effort quality of service, it is inevitable that applications must deal with errors caused by missing packets of continuous multimedia data; even with true reserved QoS and guaranteed packet delivery an application needs to deal with changes in circumstance following re-negotiations in bandwidth availability.

The encoding of multimedia content has an important part to play in dealing with such problems. Careful use of codecs can compensate for some data loss, or at least make it less perceivable by the user. At the most basic level, compression of audio and video can vastly lower the network capacity required in comparison to uncompressed data. With feedback from the transport layer (e.g. via RTCP) when network congestion occurs, codecs can be used to lower bit-rates at the expense of quality, although there is a threshold authors and users will not tolerate dropping below; indeed use of video over the Internet in recent years has been primarily constrained by bandwidth [3].

Without going into too much codec-specific detail, this section gives an overview of some of the techniques that can be used when encoding audio and video for streaming.

2.2.5.1 Audio Encoding

Before it can be streamed, at some stage analogue sound waves must have been captured in a digital form (sampled and quantised). This now digital data must be mapped into network packets, but since packet retransmission is undesirable, the technique used to allocate audio data to a particular packet can be used to offer some resilience to packet loss:

- Given temporal integrity during playback, any loss in data is transferred to a loss in sound. If the loss of sound prevents correct interpretation of noise it becomes highly noticeable to a user. For instance, speech is split into phonemes, which are 30 to 50 milliseconds long. Since the probability of losing multiple packets in a row is relatively low compared to the probability of losing a single packet [26], it is desirable to packetize the data into durations of 30 milliseconds or (preferably) shorter so that a single packet loss will go largely unnoticed. A simple 'trick' to maintain apparent quality during any gaps is to fill that time with either random noise (which is less noticeable than silence) or repeat the previous packet (a brief stutter).

- This process can be extended through *sample interleaving*, where a single packet does not contain a contiguous section of sound. Instead, a section of time is divided into several consecutive bundles of sound, each of which in turn are split into several smaller intervals. The first packet contains the first intervals from each of the bundles, the second packet all the second intervals, and so on. If a packet is lost, then the user suffers several very small breaks in audio, rather than a much more perceivable packets loss in one moment. This technique is used in the Real Audio codec (figure 2.3).

- *Forward Error Correction* (FEC) provides a similar result to continual retransmission, by adding information about the previously sent packet in each packet that is transmitted. Although this avoids the latency overheads of transport layer retransmission, if every packet has to also contain the

data of the previous our bandwith requirements are almost doubled. FEC becomes more viable because the extra correction data will not normally be needed, and so this can be encoded with use of a lower quality, but higher compression, algorithm (e.g. a 64 Kbit/s PCM stream could include a 13 Kbit/s GSM FEC stream in a 77 Kbit/s total stream). Due to the packet loss characteristics of the Internet [26], a relatively small increase in bandwidth requirements can recover up to 90% of packets lost by the network, using FEC [84]; this obviously comes at the expense of an increased latency of the length between packets being corrected for.

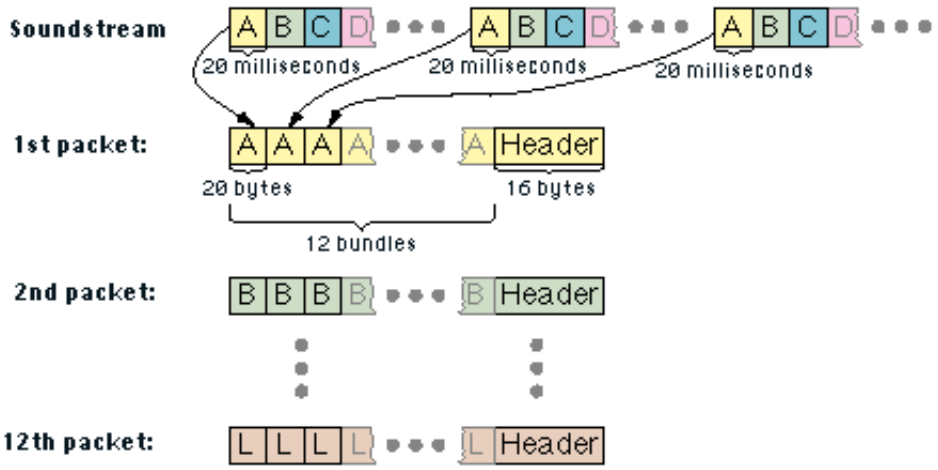

FIGURE 2.3: Sample Interleaving in Real Audio
(from [57])

2.2.5.2 Video Encoding

Video, as a series of two-dimensional image frames, can easily become very large in size. Increases in resolution, frame rate, or pixel colour depth are instantly scaled by the other factors; all of these values are typically large. Thus images and video are normally compressed to a point where they are manageable for storage and streaming, while still maintaining image quality.

The basic compression applied to MPEG and H.261 encoded video is similar in technique to those used by the JPEG image format. JPEG [142] can achieve a 10

to 20 compression factor through compacting the original information using both a more efficient encoding, and removing information imperceptible to the human eye. MPEG and H.261 employ these techniques within each frame, where it is know as an intra-frame encoding. MJPEG, or motion JPEG, *only* encodes video as a series of JPEG frames, and so requires a large amount of bandwidth even at a low frame rates.

Several further compression techniques can be used to take advantage of features unique to video:

- The sequences of of images that form video often exhibit *temporal redundancy*. In other words, there is a similarity between one frame and the next, since for a particular view the background will remain similar while major change is due to movement of objects in the foreground. By dividing the image into blocks, some blocks will stay the same from one frame to the next (those that cover the background), while others will appear to move position within the overall image (those containing the foreground). Only when a block changes above a threshold, identified using *motion detection*, does image information need to be transmitted; furthermore, instead of sending the entire block, *differential coding* can calculate a delta of change between the *target block* in question and a *reference block* from the previous frame, which can be compressed and sent with even smaller data.

- *Motion compensation*, or *predictive coding*, is used to find the best reference block for a given target block, in the process between motion detection and differential encoding. It determines where to find a block in the frame before or following, and so can produce a vector describing relative movement within the image between frames. Motion compensation is computationally expensive because a target block must be compared with many potential reference blocks, and so is usually restricted to blocks which are spatially close. It is also susceptible to packet loss, since a predictive vector for a target block is useless if the reference block was lost.

H.261 was designed for real-time encoding, and only uses forward predictive coding. MPEG, on the other hand, achieves higher compression by also using backward prediction and interpolation (bi-directional prediction), which are applied through different types of frame:

- I-frames are temporally independent, and are only encoded using intra-frame techniques
- P-frames are forward predicted from the immediately preceding frame, which might be an I-frame, or another P-frame, e.g. the sequence I-P-P-P-P-P-I-P-P-P-P-P.
- B-frames are bi-directionally predicted, or interpolated, resulting in even higher compression rates. They are coded from both the preceding I or P-frame and the following I or P-frame. The use of backward prediction requires that frames must be encoded out of order, since a frame being backward coded must wait for the following frame to be coded first. e.g the sequence I-B-B-P-B-B-P-I-B-B-P-B-B-P.

Loss of a packet is particularly damaging to the MPEG encoding, since the following frames are likely to be derived from it in some manner and full correction cannot occur until the next I-frame is received. Several techniques can be used to try overcome this problem [120], ranging from retransmission, FEC of B-frames, and multicasting the different frame types in separate groups.

Another approach is to decrease the time between I-frames. This is taken to the extreme of a single frame in MJPEG, and at the expense of bandwidth. Intra-H.261[98] transmits independent I-frames, but uses *conditional replenishment* to only intra-code blocks which have changed above a threshold between frames, so that only blocks within a frame are affected by dropped packets (it also transmits all blocks at a very low rate to ensure all blocks are updated eventually). While offering greater resistance to packet loss, it does so with greater bandwidth requirements than inter-frame (e.g. MPEG) encoding.

2.2.6 Signalling and Control

The preceding sections have introduced various facets of distributed multimedia, describing quality of service, transport and media encodings somewhat in isolation. To support multimedia applications, such as Internet Telephony (Voice over IP), these components must be brought together and used in a consistent framework, with standards to define the "glue" that allows them to do so.

In constructing such frameworks for packet-switched networks (e.g. the Internet), it is desirable for the design to provide at least the *functionality* of a traditional PSTN (Public Switched Telephone Network, i.e. traditional land-line based telephone networks in use today), despite the divergence in architecture which inevitably arises from a fundamentally different network layer. Indeed, Internet Telephony has numerous features which deliver advantages over a PSTN [131]:

Network transparency when transferring multimedia data. There is no need to change network infrastructure to support media types other than point-to-point voice, such as video and shared applications, nor to support multi-point calls, either through a centralised architecture or network multicast. Quality is adjustable by negotiation of media type (including bit rate and compression values) between end systems, rather than a level intrinsic to the network. The signalling and management of different media and call types has minimal variations which can easily be accommodated in the framework design to allow for future extensibility.

Common interfaces can be built upon this common infrastructure, both at the end system and user level. Distribution applications (e.g. radio, TV) and communications applications (telephone, fax) are no longer segregated and distinct, and can benefit from a common and enriched user interface. Call signalling and management is performed by end systems where a user can benefit them, whereas PSTNs restrict much of their signalling control to the core network.

Identification of end points, the service type, charging/billing entities, and

carrier, are conceptually separated. By comparison, a PSTN address (i.e. phone number) must combine all these, either explicitly or by implication.

Interaction with an Internet Telephony framework is provided by a further layer of signalling and control architecture, which must [48]:

- provide a consistent interface to allow the set up, management (e.g. transfer, hold, multi-point conference), and tear down of connections.
- scale to support a very large number of connections.
- support management features such as admission control, logging and monitoring.
- through a consistent and well defined interface to these features, allow interoperability between different implementations.

There are two widely adopted standards for multimedia communications systems: the H.323 suite of standards and protocols [90], developed by the International Telecommunication Union (ITU); and the IETF IP telephony stack, in which the "glue" is provided by the Session Initiation Protocol (SIP) [122]. Figure 2.4 shows a summary comparison. Both use RTP/RTCP to transport media flows, thus their differences are mostly in the way they handle signalling and control.

H.323

H.323 consists of a series of recommendations for "Packet Based Multimedia Communications Systems", the constituents of which are:

H.225.0 call establishment and signalling, Remote Access Service (RAS) for gatekeeper communication, call setup, packetization and synchronisation of media streams.

H.235 security protocol, for authentication and encryption.

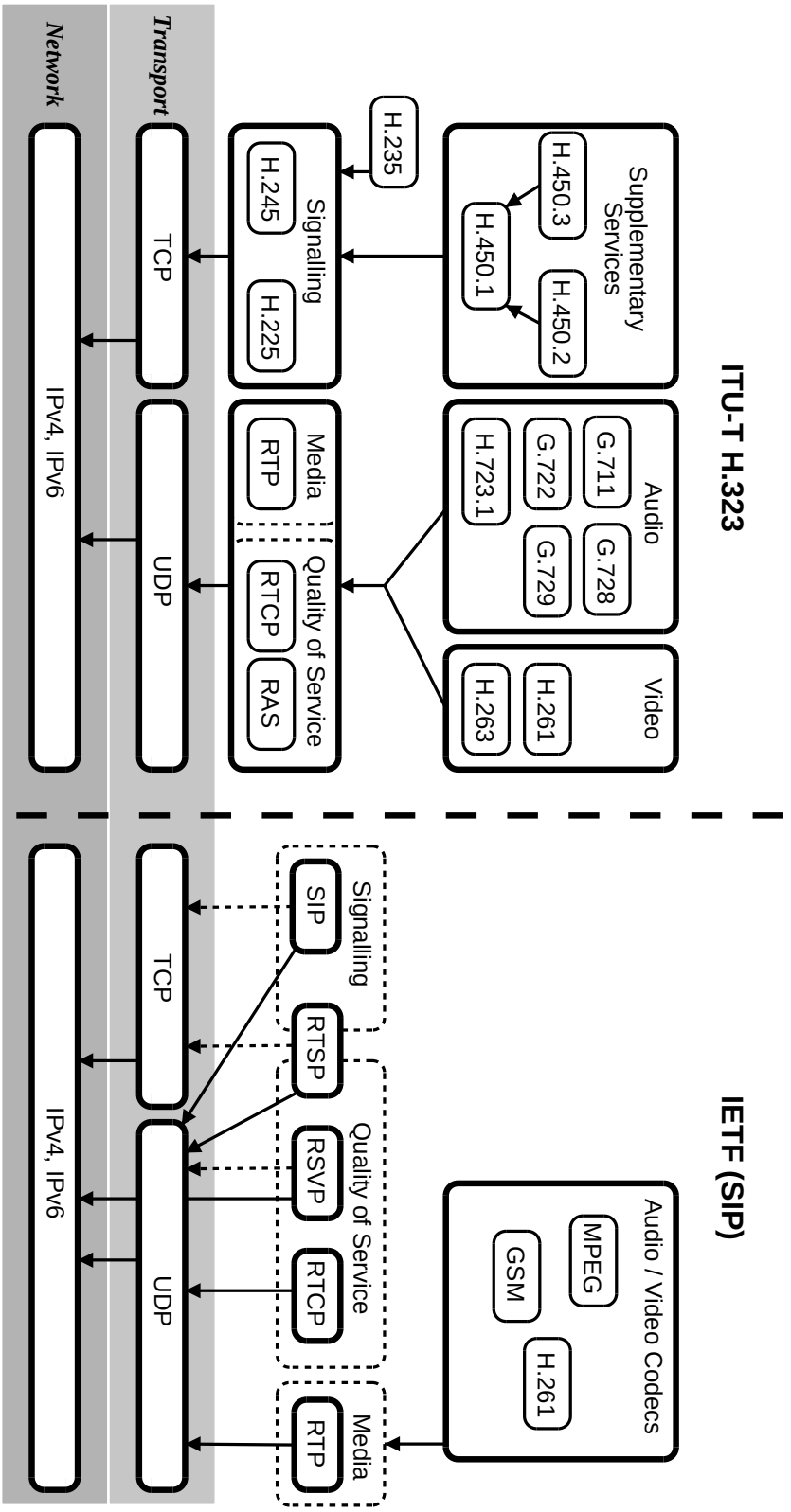

FIGURE 2.4: The H.323 and IETF Protocol Stacks

H.245 capability management and control.

H.450 supplementary services, e.g. H.450.2 for call transfer, H.450.3 for call diversion, H.450.4 for call hold.

H.246 for interoperability with legacy circuit switched services.

H.332 for large conferences.

H.26x video codecs, including H.261 and H.263 (see section 2.2.5).

G.7xx audio codecs, including G.711, G,723.

Four major network components are explicitly defined:

Terminals are client endpoints which provide real-time communication with other H.323 components, and must support H.245 and H.225.

Gateways provide bridging functionality to circuit switched networks, e.g. PSTNs.

Gatekeepers perform address translation (aliasing of phone numbers to network addresses), admission control, bandwidth control, and zone management. When present, other H.323 endpoints register with a gatekeeper, which manages signalling for that endpoint prior to, and during, a call.

Multipoint Control Units (MCUs) are used to support conferences that involve more than two endpoints. Endpoints connect to the MCU, which provides audio/video switching and mixing.

As can be seen, H.323 is a large and complex specification; much of the size is a legacy of its foundation upon earlier ITU multimedia protocols, such as H.320 for ISDN, and H.324 for circuit switched PSTN terminals (i.e. videophones). There is no clean separation between the components in the H.323 family, and many services require interaction between them. H.323 messages are transmitted in a binary encoded format based on ASN.1 and the packet encoding rules (PER).

The Session Initiation Protocol (SIP)

The IETF based multimedia communications protocol stack has generally taken a more simplified approach, using existing protocols where appropriate and when new protocols are required moulding them in the image of established standards such as SMTP and HTTP, as befits its more Internet-centric heritage. As with H.323 RTP/RTCP is used to carry media flows, however the native RTCP feedback and conference control mechanisms are fully utilised (compared to their duplication and extension in H.323/H.245).

Signalling and control is performed by SIP, which is a client-server protocol. Endpoints are known as User Agents (UA), and comprise of a:

User Agent Client (UAC) which issues SIP requests.

User Agent Server (UAS) which responds to these requests, usually as a function of user interaction.

SIP clients communicate with servers using a simple, HTTP like, protocol where the client calls methods on the server to signal and control calls (e.g. INVITE, CONNECT, REGISTER, ACK, BYE). The SIP protocol is text based, and the header is largely self-descriptive, containing a From address, a To address, and a Call-ID unique to that session. Users are identified using an email like identifier of the form *user@domain*, *user@host*, or *phone-number@gateway*; these identifiers form part of a SIP-URL, which is prefixed with *sip:*.

A basic SIP call can take place between two stand-alone UAs (the UAC of one connects to the UAS of the other, and vice versa), however full IP telephony functionality is only available when further *network servers* are used. The protocol does not distinguish between a UAS and a network server, they only differ in function:

Redirect servers informs a client of the address of the next hop server (which might be a UAS or another network server), so it can connect to that server directly.

Proxy servers receive SIP requests and forward them to the next hop server; they may be stateful or stateless.

Registrars receive SIP registration messages from User Agents, and store the user location (a SIP URL and the associated IP address) information in a non-SIP location store.

2.2.7 Computer Supported Co-operative Work

In contrast to the systems perspectives of previous sections, the field of Computer Supported Co-operative Work (CSCW) takes a more user-oriented approach. It was born out of work in both Human Computer Interaction (HCI) and Computer Mediated Communications (CMC) in the context of supporting activities within and between *groups* of users [79], and as such combines elements of the social, as well as computer, sciences. Whether a particular system is considered a CSCW (or *groupware*) application will often depend on the setting and means by which it is used.

Co-operation is described as the most elaborate of four levels of human interaction [24], placing higher demands on the tools which support this activity:

1. Informing (no acquaintance): Information is communicated anonymously through the mass media or local resources such as bulletin boards or newsletters.

2. Co-ordinating (some acquaintance): Not necessary to have common work goals, although common interests and organisational affiliation are likely resources.

3. Collaborating (working relationship): Participation in the same process, such as preparing a memo, processing a form or developing software.

4. Co-operating (goals are common): Sublimation of individual goals in favour of the team's goals.

		Time		
		Same	Different but predictable	Different and unpredicatable
Place	Same	Meeting Facilitation	Work Shifts	Team Rooms
	Different but predictable	Video conferencing	email	Collaborative Writing
	Different and unpredictable	Intereactive multicast	Bulleting boards	Workflow

TABLE 2.1: Space / Time classification of co-operative work (from [79])

As we have seen, there is a large body of multimedia systems research into distributed solutions and dealing with temporal issues; these dimensions are also highly relevant to CSCW researchers, in terms of where and when the elements of co-operation take place [59]. This can be extended to take into account the predictability of time and location (table 2.1, including examples of applications).

CSCW systems can be further classified by location along one axis, and by *synchronicity* of interaction on the other - asynchronous for creative brainstorming tasks, synchronous for planned and prescriptive tasks. Placed within this classification are the four classes of CSCW systems [121]:

1. Messaging Systems
2. Computer based Conferencing (multimedia and real-time)
3. Meeting Rooms
4. Co-authoring and Argumentation

CSCW should not, however, be defined merely in terms of the techniques being applied [15]. In drawing together many aspects of computer science and other fields, it is united by the requirements of supporting co-operative work rather than a technology driven approach. Nevertheless, it is worth being mindful of the issues faced by the CSCW community, and the analogy with parallel issues in the other areas discussed in this chapter and thesis, most pertinently:

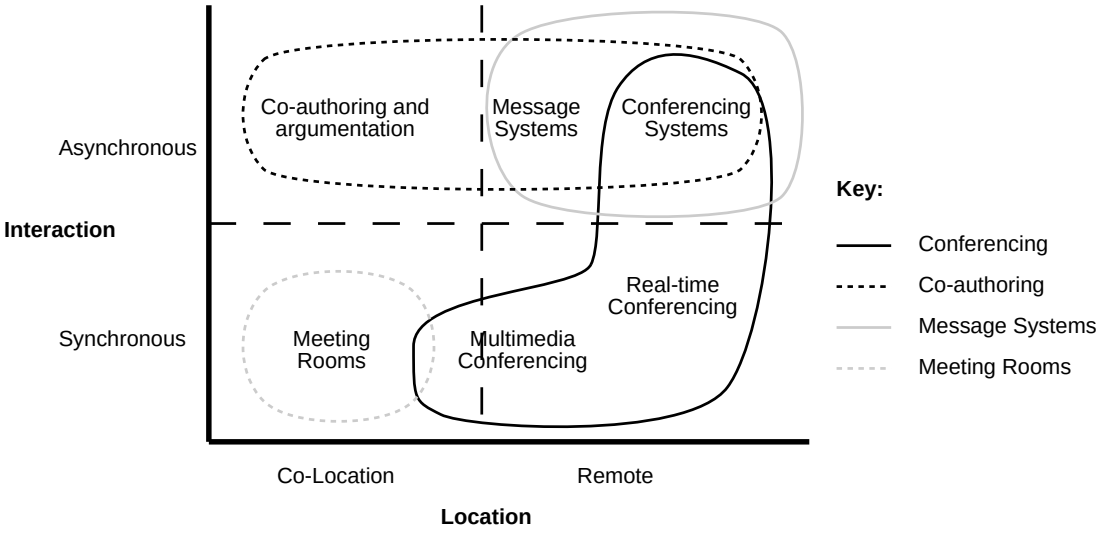

FIGURE 2.5: Interaction / Location classification of CSCW systems (from [121])

- distributed multimedia applications and conferencing, including the combination of group communication and synchronisation (real-time distributed multimedia, section 2.2), and authoring, presenting, and interacting with the media (temporal hypermedia, section 2.3).
- shared information spaces, both in the possible distributed nature of the information, navigation of structure within and between the information (hypermedia, section 2.1), and management and structuring of knowledge (metadata and knowledge, section 2.4).

2.3 Temporal Hypermedia

2.3.1 Hypertext and Temporal Media

In the previous section we examined how continuous real-time multimedia can be transported across a distributed system in a controlled manner. Once this content reaches the user it cannot be presented effectively within a hypertext environment unless that system explicitly incorporates time, and many hypermedia models and systems do not.

The requirement for temporal support can be expressed from three perspectives [86]:

1. When an author creates a hypermedia document using multiple media elements, the temporal arrangement of these elements is important so as to reflect the chronological development of the presentation.

2. A hypertext system designer must extend the hypertext model with concurrent multiple media streams, but traditional models are unable to explicitly express temporal relationships (links).

3. A multimedia system designer has support for multiple media, including temporal types, but lacks any linking between them. Rather than linear presentation, navigation is required: within the timeline of a single linear presentation; between a web of multiple linear presentations; and within and among non-linear multimedia presentation.

Several projects have been developed to enable the use of audio and video within hypermedia.

The Firefly Document System [35] created documents from media items, temporal synchronisation constraints, and lists of display controlling operations. Time based navigation is available, although these links were resolved at display time, rather than during authoring. The entire document was "compiled" using a scheduler component, which solved the temporal constraints to create a single document object, and it was this inflexibility which made it unsuitable for more general situations involving distributed multimedia.

A Sound Viewer [75] was developed for the Microcosm system which presented links associated with audio media through a visual interface. A timeline of links which scrolls in relation to the temporal position of the audio element is used for both authoring and resolving links, and this tool has since been extended to stream the audio source from an RTSP server. More recently, the Arakne Environment has been extended to provide simple linking to, from, and within

temporal media on the WWW. This Mimicry system [30] uses a Java plug-in to enable the WWW browser to interact with the Hypermedia system.

Meanwhile, the broadcast entertainment industry is increasingly interested in providing content management and navigation systems to supplement the continuous media services they provide. TV Anytime [61] has been driven from a more user-oriented perspective, and lacks the sophistication of full hypermedia services. It does, however, provide a set of navigational technologies to integrate with both temporal content (stored locally on a user's device from broadcast) and external Web based data. Also used extensively in broadcasting, for example in digital TV set-top boxes, MHEG [63] is an ISO standard which defines multimedia as collections of related objects for interchange and synchronised presentation, and a procedural language to describe the interactions between the objects and their presentational semantics.

These tools have enabled the use of temporal hypermedia in several different ways; the rest of this section will give an overview of the most important models and standards in this field: HyTime, the Amsterdam Hypermedia Model, and SMIL.

2.3.2 HyTime

The Hypermedia / Time-Based Structuring Language, HyTime, is an SGML-based standard primarily concerned with describing relationships between different parts of documents [40]. It is not an SGML DTD, and does not provide a single document architecture; rather it extends SGML with a standard set of abstract components used to build document architectures, and specifies clearly how these architectures should be coded.

HyTime greatly enhances the coarse SGML element labelling model of referencing objects, using sub-addressing techniques to easily narrow in on any part of a document, and in a form which can be returned by a query. It also has complex and powerful hyperlinking facilities, and perhaps most importantly, the

ability to precisely represent intricate temporal information about objects.

A combination of modules can be used as needed [40]:

- The base module is always used and provides SGML constructs for object representation and addressing.
- The measurement module is used to address document objects using abstract measurable domains such as time and space.
- The location address module provides the extended object referencing capabilities of HyTime, allowing addressing and sub-addressing by name, position and query.
- The hyperlinks module is used to create link objects between and within documents and document objects. Standardised interfaces are available for links including contextual links (clink), independent links (ilink), aggregate traversals (agglink) and query links.
- The scheduling module uses finite coordinate spaces to position events which occur along that measurable axis.
- The rendition module is used to describe how an object can be modified in an event, and how events can be projected from one coordinate space to another.

For example, a finite coordinate system could be used to represent a timeline, and objects within a presentation could be mapped onto the timeline at appropriate moments.

Although HyTime provides a very thorough method for describing document features and relationships (such as hyperlinks), it does not define what these features mean (e.g. when is a link followed, how is it activated?). It also lacks any definitions for the presentation specific aspects of documents, and as such is unsuitable as a complete temporal hypermedia standard.

2.3.3 The Amsterdam Hypermedia Model

Although including temporal multimedia within the structure of a hypertext system seems like a simple idea, most hypertext models cannot successfully incorporate complex collections of dynamic information. The Amsterdam Hypermedia Model (AHM) [85] attempts to present a framework that can be used to describe the basic features and operations common to hypermedia systems. It extends the Dexter hypertext model to include notions of time, high-level presentation attributes and link context, and while the model is based on a large number of detailed requirements [84] this section will focus on those related to the needs of temporal media.

The AHM extends the **atomic components** of Dexter by including temporal duration in the presentation specifications. This allows **composite components** to be used to build a presentation structure rather than simply collect together related components, and thus the composite becomes the main mechanism for supporting temporal relationships in the AHM [85].

A composite can either display all its child components at once (a parallel composite) or only show at most one at a time (a choice composite), and has no need for a duration value itself since this can be calculated from the durations of its child components. Each child within a composite can be given an explicit start time, and this allows coarse-grained synchronisation between them. Finer, more powerful synchronisation is achieved using **synchronisation arcs**, or **sync arcs**, which are constraints the run-time system should enforce upon the behaviour of two or more components.

While not a navigational link, the sync arc has endpoints representing the source and destination components. The timing relation is included as a target time with allowable margins either side and a synchronisation type. This type expresses whether the arc is relative to the start, end, or an offset into the component, and defines the synchronisation as hard (the constraint must be met and the application stop) or advisory (the application should continue

regardless). With these features a sync arc can express all the temporal relationships needed to ensure a synchronised presentation of components.

In the AHM components that are to be shown together can be clearly separated into composites, and within the composites relative ordering and detailed timing constraints can be expressed [85]. Highly complex temporal presentations can be represented using the AHM; these ideas can be further expanded by introducing the concept of presentation time (see figure 2.6).

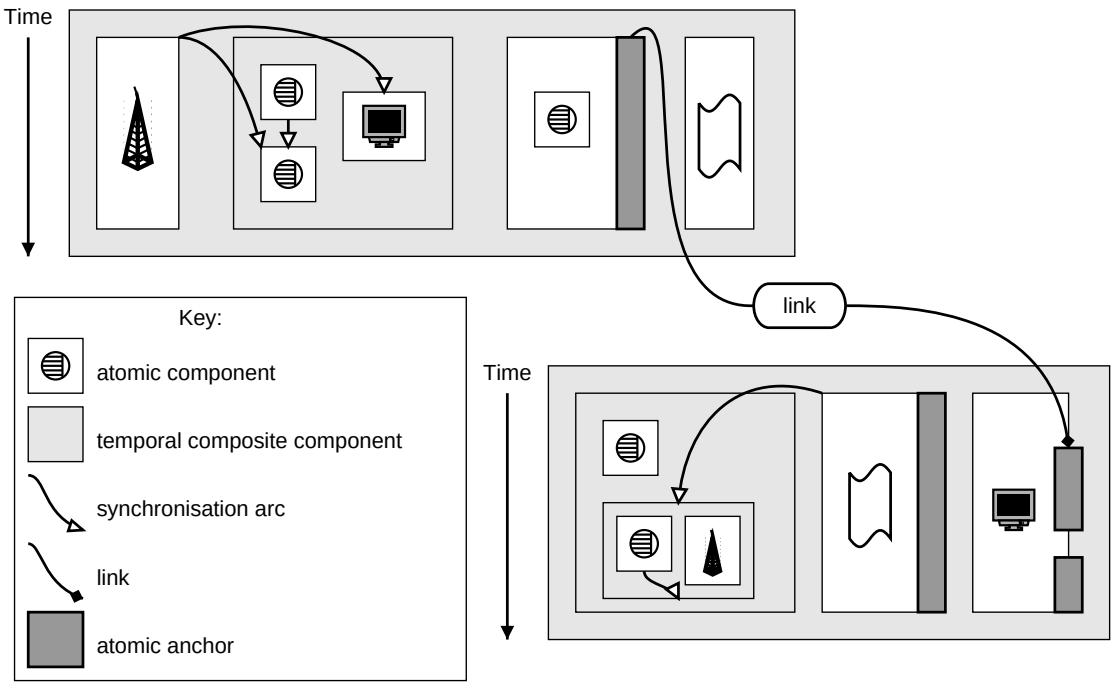

FIGURE 2.6: Amsterdam hypermedia model overview
(from [84])

Presentation time [86] is the timing of the individual parts of a presentation and the temporal relations among them, and refers to timing from the perspective of the reader experiencing the document as it is played (where a document is the storage representation of the presentation, whether a single file, multiple files or database). Presentation time can be further split into four sorts of time, each of these having its own time axis which can be used for synchronising the control of other events, such that an event of one time type can trigger an event in another time type.

Media element time is the temporal length of the segment of media played in the presentation, and is an inherent property of that media element. Text, still images and live broadcasts have an indefinite duration.

Document time is the time assigned by an author for an element within the presentation which is then stored in the document.

For atomic components the document time will loop/stretch or clip/shrink the element in relation to its media element time, corresponding to the event projections of HyTime.

For composite components the document time is dependent on the synchronisations specified between the child components. If the synchronisation is defined as soft, then if content is delayed at runtime the rest of the presentation should continue; if hard, the rest of the presentation should pause until the content returns. Sync arcs specify the relative start times of child components, but if one component has an indefinite duration then a second sync arc can be used to instantiate the duration (scaling the indefinite duration to fit).

Rendered Time describes the time which is resolved after a choice has been made to select one of multiple alternatives for a particular media element. These choices may be offered to adapt the presentation to user preference or system hardware. Up to rendered time each combination of alternatives would produce a different presentation time.

Runtime is the time it takes, in real time, to play the presentation and includes any interaction the user may make with the presentation. These interactions may include pausing, playing, fast forwarding or reversing and traversing links within the presentation and to other presentations. The situation is complicated further when the presentation is made up of more than one temporal composite, where one rendered time line continues while the user navigates through others.

It is argued [86] that these aspects should be incorporated into a document model of hypermedia in which time becomes a first class object, just as links did

in hypertext.

2.3.4 SMIL

The Synchronised Multimedia Integration Language, SMIL [37], is a standard for presenting multimedia on the WWW that takes on board many of the issues presented by the AHM. Part of the W3C vision for a next-generation Web, it is an XML based format where the syntactic markup is used to position sections of the document with respect to time, rather than the normal use of markup for layout (e.g. XHTML) [124]. SMIL specifies a presentation in six parts:

Content - Media items represented in SMIL may exist in spatial and/or temporal domains, where they may be continuous or discrete. A media element, of which there are several specific element types, will be marked up with a source and type for the media itself. If only a segment of that source is being used the section will be bounded by variables in the relevant dimensions (space and time).

Spatial Layout - Spatial regions are defined for the presentation space, and media elements then reference the identifier of the region they will appear in (at the relevant time). The regions are laid out using a subset of CSS2 (Cascading Style Sheets); media elements are then placed in the region in a manner prescribed by the author (e.g. scaled or cropped). Elements also have a Z-axis (depth) specification to allow for overlapping regions.

Temporal Layout - Media elements either have an intrinsic duration or can be allocated an explicit duration; this duration can be extended by repeating the content. Temporal elements can be grouped so that siblings are played in parallel or sequentially, and these groups can be nested and mixed amongst each other for flexibility. Within the groupings start and end times can be given either explicitly or relative to other elements; synchronisation between elements can be hard or soft.

Links - Linking can be specified to external documents, within other SMIL elements or to anchor points within other document types embedded in the SMIL presentation. A linked external presentation either replaces the original or can be displayed separately; while it is viewed the original presentation will either pause or continue running simultaneously. Linking to within an SMIL presentation can resolve to temporal and spatial sections of an element; temporal links will effectively fast-forward the presentation to that point in time.

Alternative Content - To allow for differences in network connections, hardware capabilities, and user preferences (such as language), test attributes can be evaluated to choose a particular child from a switch element. This allows a presentation to be tailored for a particular user's needs.

Semantic Annotations - Metadata elements can define the properties of a document or individual element. This information typically includes element titles, author details, expiry times and keyword lists, but does not include more sophisticated metadata techniques to describe content types.

Using XML tags and attributes an author can mark up a document using these specifications for many different purposes. Despite some limitations, SMIL has been successfully applied to create differing types of multimedia: enhanced "infotainment" multimedia, accessible multimedia and conceptual multimedia art being three examples [123]. Its acceptance as a W3C standard has led to increased use of the language across the Web and support for SMIL presentations in a number of web-based client programs. Parts of SMIL are also used within other standards to describe temporal facets; for instance the Scalable Vector Graphics (SVG) [31] format uses SMIL for timing events to produce animations.

SMIL 1.0 has a relatively basic feature set and was seen as a first step to developing more functional temporal hypermedia systems for the WWW. The continued development of the SMIL standard through to version 2.1 Bulterman

et al. [38] embraces many of the results from recent hypermedia research [125]. The introduction of atemporal compositions (from the AHM) allows true non-linear presentations where temporal elements can have a semantic relationship defined between them without the need for a predetermined temporal relationship [86]. Broader synchronisation will be possible between non-siblings, and a more generalised event model and integrated scheduling model allows link traversal to be treated as a special case of a more general event type. The feature set of SMIL has also been extended in version 2.0, to provide a richer set of multimedia features, such as animation and transition effects, and more recently the addition of profiles for mobile devices in 2.1.

Of further importance, SMIL 2.0 also introduces profiles and modularisation, in an evolution similar to that from HTML to XHTML. The standard has been separated into ten functional areas (Timing, Time Manipulations, Animation, Content Control, Layout, Linking, Media Object, Metainformation, Structure, Transitions), and then further partitioned into 45 *modules*, spread between the functional areas; all of these modules are XML compliant within the SMIL namespace. Firstly, this means that SMIL can be tailored to suit different environments, while maintaining compatibility, by prescribing set groups of modules, called *profiles*. For example, a language based upon the SMIL 2.0 Basic profile need only include 10 modules, and would be more suitable for low resource display devices. Secondly, this means that the SMIL 2.0 modules can be reused within other XML languages, such as SVG using the SMIL animation module.

Despite all its success in brining hypermedia constructs and principles into more widespread use, SMIL remains a presentation oriented language, and is mainly of use in describing how a document should be shown to a user. It does not deal with issues regarding how temporal data should be retrieved and dealt with in the steps leading up to presentation and cannot be considered as a complete hypermedia system; in particular it lacks the capabilities for fully integrated authoring.

2.4 Metadata, Information, and Knowledge

2.4.1 What is Metadata?

Metadata is ancillary data about other data, and as such is data itself; it can mean many things to many people. In general it refers to any data used to aid the identification, description and location of some other electronic resource. Many forms of metadata exist; some are very simple while others are quite complex and richly featured.

The definition of what metadata is and what it can represent is very broad, and while the generation of metadata is no simple task and should not be overlooked, it is the method of interchange of metadata between systems and applications that is the focus of most research. The whole point of metadata is to aid the understanding of other data, so there must be a way to decode the metadata into useful information or it becomes as useless as the data it is augmenting. Without common structures and standards for metadata there can be no interchange and translation between systems; without consistent interpretation metadata has no value.

This section gives an overview of how metadata can be used and structured, with particular reference to the three relevant areas already studied in this chapter: how metadata can be used to describe and add structure to multimedia; how metadata can be used to mark up and describe elements in hypertext systems, including the Web; and how the Semantic Web uses metadata to structure information into knowledge.

2.4.2 Metadata and Multimedia Content

A common metadata format for temporal multimedia could be considered of greater importance than one for the (mostly text-based) WWW, since it is even more difficult to generate metadata for media such as audio and video without

commonly recognised descriptions. In the multimedia realm metadata has many attractive potential uses: semantic searching, indexing, retrieval and filtering of multimedia databases; image understanding for intelligent vision and surveillance; and conversion between media (speech to text etc.).

Multimedia content, such as audio and video, is generally very *detail rich*, but *structure poor*. In other words, it contains a huge amount of data - consider all the individual pixels in a frame, and all the frames that form a video - but very little inherent structure to make sense of that data. In streaming media, the only consistent structure is time, in the form of frames or samples, and this is disjoint from the information conveyed when the media is viewed (although time is a useful axis to index metadata against).

Nack and Lindsay propose four fundamental ways of describing multimedia data that could form useful metadata Nack and Lindsay [103]:

Medium-based descriptions are of the medium in which the data is expressed, such as the sampling rate or the camera's focal depth.

Perceptual description breaks the media into perceptual objects such as colour, texture or sound.

Physical descriptions are of features that do not correspond to human perception, and can be easily derived from raw multimedia data. Examples include "level" and "frequency" (compared to perceptual "loudness" and "pitch").

Transcriptive descriptions represent a reconstruction of the real world structure as captured by the data. For example, a musical score can represent audio data; or a dialogue transcript for a television recording.

For any particular segment of multimedia data several of these description classes can be used to give different views of the same data. Any multimedia metadata standard must not only accommodate these independent viewpoints, but also make them complementary rather than mutually exclusive [103]. There may also

be a need for an architectural description to formalise the structure of the other description classes, the data they represent, and relationships between them.

The Moving Pictures Expert Group (MPEG) of the International Organisation for Standardisation has produced several standards for the coded representation of temporal audio and video, with varying levels of metadata support:

- The MPEG-1 standard for storage and retrieval includes a mechanism for multiplexing a data stream (which could be metadata) into the MPEG-1 stream, but does not prescribe how to format this data (which has led to proprietary, incompatible, implementations).
- MPEG-2, the digital television standard, is an extension to MPEG-1 that utilises a higher resolution. It offers limited additional metadata support through a structured information block in its header, which can be used to encode copyright and access information.
- MPEG-4 is a standard for the production, distribution and content access of multimedia, and is designed to be applicable to a wider range of fields than the earlier standards. It still deals with streams, but subdivides audio-visual content into objects. Metadata can be attached to these objects, but again there is no standard structure or format.

The Multimedia Content Description Interface, or MPEG-7, is not a standard for transmitting or storing multimedia data. Instead, it aims to standardise a core set of quantitative measures of audio-visual features (*Descriptors*) and structures of descriptor relationships (*Description Schemes*) [103]. MPEG-7 will also introduce the *Description Definition Language* (DDL) to specify new Description Schemes, which has the same aims for multimedia metadata as RDF does for the WWW, and uses the XML Schema language as its basis with a view to future interaction with non-MPEG-7 metadata. RDF was originally considered unsuitable for inter-operation of multimedia metadata since it lacks explicit linking mechanisms to spatio-temporal sections of data and has limited data

typing [104], there have since been efforts to express MPEG-7 as an ontology and encoded in RDF [88].

Since the earlier MPEG standards have mechanisms to include metadata, but no standard metadata format, it is envisioned that they will use MPEG-7 to improve their content description capabilities, although this does not preclude using MPEG-7 with other, non-MPEG encoded, media. A further standard, MPEG-21, is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities - the glue to bring actors and actions upon mixed media documents and presentations together in a standardised framework, including digital rights management elements.

2.4.3 Metadata and the WWW

The Web allows access to a vast amount of hypertext information distributed across the globe, but it can be very hard to find the type of data you require. Content retrieved from the WWW must be mostly analysed by the end user; catalogues and search engines attempt to automatically scan the Web and build metadata but the results are often inaccurate and imprecise.

As discussed in section 2.1.9, XML is the basis for many new developments on the WWW. Another foundation is the ubiquity of the Uniform Resource Identifier, or URI [20], which is used to identify resources held in the information space that is the Web. It is of little use to be able to state something about a fragment of information if we cannot identify that resource. A subset of URIs called URLs (Uniform Resource Locators) are familiar as the addresses used to access web pages, FTP sites and other web services, e.g. *http://www.example.com/testing.html#section2*. URIs are not, however, *required* to resolve to a particular document available on the WWW; they are intended to provide a consistent naming mechanism for any resource that requires reference. The URI *scheme* defines the namespace of the URI, and can restrict the use and

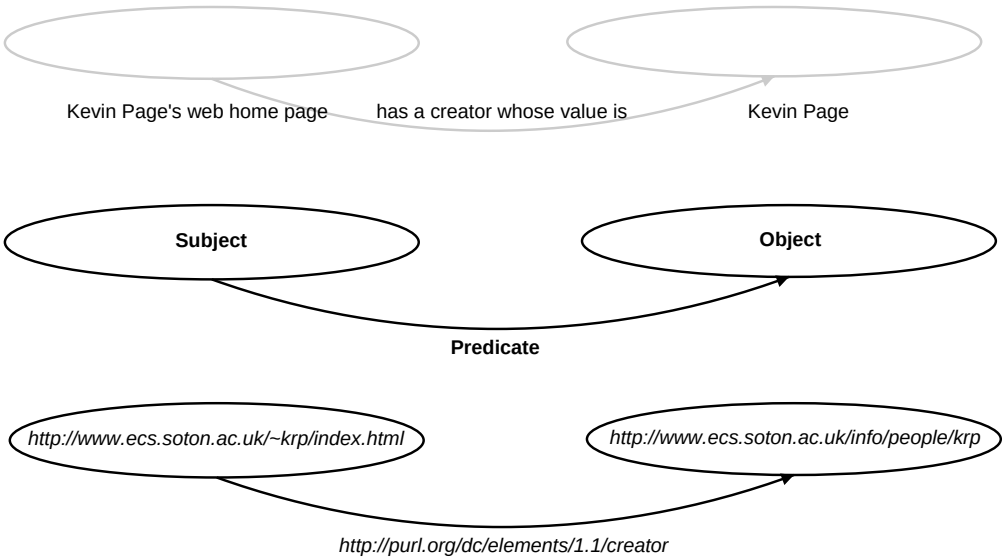

FIGURE 2.7: An example of an RDF triple

syntax of identifiers therein; often the scheme is based upon the protocol used to access that resource, e.g. *http:*, *gopher:*. Uniform Resource Names (URNs) are also a subset, which are intended to serve as persistent, location-independent, resource identifiers.

The Resource Description Framework (RDF) [94] is an infrastructure that enables the encoding, exchange, and reuse of structured metadata on the WWW. RDF does not prescribe semantics for each particular resource description community, instead it provides the ability to define new metadata elements as needed. It should not be confused with an XML file format, although its primary serialisation is an XML syntax. Its data model defines a *resource* as any object uniquely identified by a URI , and resources have *properties* which express the relationships of *values* associated with that resource. The values can be either atomic (strings, numbers etc.) or other resources (which may have their own properties) [102]. The resource the statement is about is known as the *subject*, the part that identifies the property or characteristic of the subject that the statement specifies is called the *predicate*, and the part that identifies the value of that property is called the *object*; RDF is built around creating these subject-predicate-object *triples* (figure 2.7).

Collections of properties about a particular resource form a *description*;

collections of properties used to describe resources within a particular resource description community form a *vocabulary*. RDF includes a mechanism for declaring and publishing domain vocabularies as RDF Schemas (RDFS - themselves expressed using RDF), so that RDF can support any number of descriptive requirements without needing to define them. Vocabulary semantics can therefore be understood, reused and extended, in a modular manner using the XML namespace mechanism by any system supporting RDF.

For example, the Dublin Core Metadata [144] is a simple cross-domain vocabulary designed for resource description (title, creator, description, publisher, date, etc.) on the WWW which can be expressed using RDF (figure 2.7).

2.4.4 Metadata and Open Hypermedia

Open Hypermedia and RDF can both describe the relationships between resources, but do so in different ways. RDF identifies these resources using URIs, which unlocks the large quantities of data on the WWW, but does not function as well as the more general structuring mechanisms provided by OHSs. Furthermore, with no culture of authoring on the Web, how will the metadata that RDF supports actually be created in the first place?

The Open Hypermedia Interchange Format (OHIF) has been proposed to allow the use of Open Hypermedia structures as metadata for WWW resources [78]. Using an XML DTD OHIF encapsulates the standard OHS navigational data model. Through use of an augmented web browser or an OHS (in this case Arakne), OHIF structures can be authored and presented as ordinary web resources. Representing Open Hypermedia structures as metadata is one way of adding the functionality and flexibility of OHSs to the WWW.

2.4.5 Data, Information, and Knowledge

The nature of knowledge - what it is and what it means - is a long and continuing debate of epistemological philosophy. One model, popular in knowledge management, is of a hierarchy "from data to wisdom" [4], where each stage builds upon the previous:

Data is a set of syntactic entities: patterns with no meaning that are an input to the interpretation process. In computing, these might be raw symbols, sequences of numbers or characters, or pixels in an image.

Information is interpreted data: the data has a relational connection to other data; it has meaningful structure.

Knowledge is learned information: based on reasoning over information.

Wisdom is evaluated understanding, based on judgement.

Of course, there is no inherent progression that data will become information, then knowledge - to progress from one stage to the next there must be an increase in *understanding* - and from a systems perspective in doing so there is an increase in connectedness, in *structure*, and a building of context.

2.4.6 The Semantic Web

As has been seen in previous sections, the WWW has developed increased functionality as a hypertext system, and the ability to better structure information using metadata. This work also forms the foundations for the Semantic Web [21], where the resources and information available on the WWW will be enhanced with reasoning and understanding, not just for human consumption, but for automated processing and inferencing by software agents. The strategy for the Semantic Web is broken down into several layers (figure 2.8), where the technologies of each layer build upon those previous - an instantiation of the data-information-knowledge hierarchy.

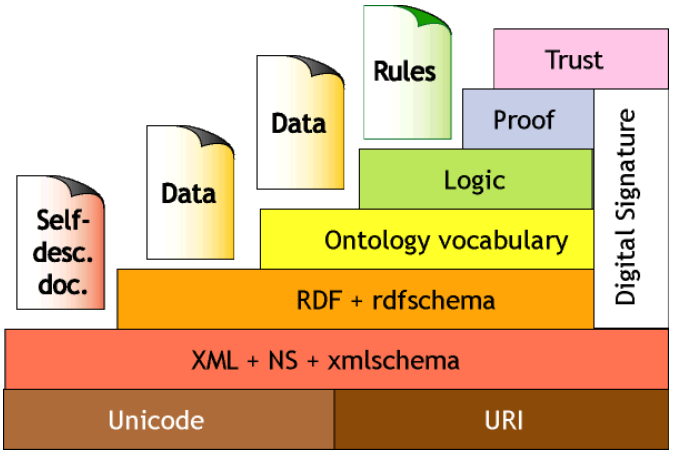

FIGURE 2.8: The Semantic Web (from [21])

The Semantic Web is not a replacement for the current WWW, but an extension, and so its basis remains the vast amount of distributed data available using common and open Internet protocols, and addressable using URIs. Information, created by structuring this data, is then enriched as the Semantic Web layers are ascended.

While some data on the web is completely unstructured, other data has some basic structure due to its encoding in XML, and further structure is held in the links between data (the hyperstructure). But this structure within and between XML documents is arbitrary; there is no more explicit meaning to it than could be automatically understood and processed by a piece of software.

RDF builds on this by providing an *assertion* layer, in which metadata is used to add meaning to a resource on the WWW. RDFS can be used to define how the metadata can be asserted.

The next layer, provided through the Web Ontology Language (OWL) [18] is needed to describe the relationships between these asserted identifiers, to document these terms and concepts (and is based upon earlier work such as DAML+OIL). The formal definition of relations amongst terms in a vocabulary is called an *ontology*, and in the Semantic Web will typically consist of a taxonomy and a set of inference rules. For example, an ontology might define a *faculty* as an organisational sub-division of a *university*, and a *department* as a sub-division

of a *faculty*. An inference rule might state that a *student* who is a member of a department is therefore also a member of the faculty of which that department is a sub-division. Ontologies also allow expressions of equivalence between terms in different ontologies - this is particularly important in a distributed environment such as the WWW where it is very unlikely that the same ontologies will be used, or even be appropriate, throughout the entire system.

The Semantic Web is very much an active research area, and the upper layers shown in figure 2.8 are still to be developed. It will be realised when we can create and use many programs that collect Web content from diverse sources, which will process - *understand* - the information, and exchange the results with other programs.

Chapter 3

Metadata in Support of Streaming Media

Chapter 2 documented the development of hypermedia systems and how they have become distributed over networks, more specifically the Internet. Multimedia systems which use temporal multimedia (such as audio and video) have also developed in distributed environments, where they demand additional support from the underlying network and transport protocols; hypermedia systems have intergrated such multimedia content through the development of hypermedia models and presentation languages with explicit support for time. Standardised metadata and vocabularies provide a framework in which to generate and manipulate information to support structured processing, which are being applied to large scale distributed hypertext through the Semantic Web.

In the first part of this chapter a lifecycle model of streaming is presented as a new way to evaluate the systems studied in chapter 2 (section 3.1). The model characterises the process media goes through as it traverses a distributed multimedia system; functionality in systems is categorised into three layers, and the provision for these layers across the lifecycle is examined.

It is asserted that, in all these systems, the structured interpretation of data is paramount in its use as hypermedia, and interpretation as knowledge (section

3.3.1); that there is a commonality in the use of various types of metadata to support and enhance distributed multimedia applications, and a generalisation of structure in metadata to which the principles of structured interpretation, such as those from Open Hypermedia, are still applicable.

After analysis of metadata in multimedia systems in the context of the lifecycle model (section 3.2), the second part of the chapter describes the deficiencies in supporting metadata through the distribution, or streaming, stage of the lifecycle. To address these issues, the notion of *continuous metadata* is introduced (section 3.3), where explicit support for time-based processing of associated metadata is extended back across the network from presentation towards production.

The aspects of the metadata layer that continuous metadata is required to embrace are investigated, and finally the use of continuous metadata is illustrated in several motivational scenarios (section 3.4).

3.1 Components of Distributed Multimedia Systems

In section 2.2.1 it was seen that, from the users' perspective, distributed multimedia can be classified into three types of application: presentational, conversational, and combined [80].

From a systems perspective, the basic elements needed to support these applications can then be categorised as [5]:

1. Explicit support for continuous media.
2. Quality of Service.
3. Synchronisation.
4. Group communication.

Fulfilling these conditions poses various obstacles at all levels in a multimedia system, from basic network support for quality of service and multicast, through protocol support for encoding and transporting high bandwidth, time sensitive media, to synchronised presentation and navigation. Chapter 2 described research and solutions associated with supporting these requirements.

3.1.1 The Lifecycle of Streaming Media

We will now further analyse the properties and support of distributed multimedia applications through consideration of a 'lifecycle' of the streaming media at the heart of these systems.

Inherent in its distributed nature, the media within an application must be carried across a network (streamed). We can broadly break the streaming process into three stages, which form one axis of the lifecycle:

1. *Generation.* Setup and signalling frameworks are used to determine the endpoints for network transportation, which might be point to point or multicast. Quality of Service requirements and limitations must be determined, negotiated and assigned; output of the media must be controlled and adapted to allow non-linear playback when requested. The media is encoded using a suitable codec for its type, to reduce the required bandwidth and provide improved network efficiency and resilience, and is annotated with descriptions about its type and content using metadata.

2. *Transmission.* The network transports the media data between endpoints, and control messages provide QoS feedback which may be required to limit or increase generation accordingly. This stage of the lifecycle, in particular, is indicative of *streaming* media: a continuing, perpetually repeating, process which is explicitly supported by the media encoding and network transport and distribution.

3. *Presentation.* The receiving endpoint reassembles the media stream from

> the network transport protocol, then decodes the multimedia into a format suitable for the user's display device. Any loss of data is overcome through a combination of the codecs resilience to data loss and the presentation application adapting. An unworkable stream of data is rectified by sending control messages back across the network (to rectify output by the generation stage). Metadata is used to both provide details about the media content and type so the display application can adapt accordingly, and to prescribe the layout, presentation, and synchronisation, of a piece of multimedia in relation to other media segments. Metadata describes any hyperstructure for navigation between and within elements; these may lead to switches in media which necessitate signalling and control requests being relayed back to the generation stage, where the change can be applied.

These and other requirements are placed upon the system at each stage, as per *Adcock et al* [5].

It is important to distinguish between this lifecycle and a straightforward linear temporal axis. Although there is, and must be, a temporal context to the lifecycle, the complete sequence is continuous. The three stages are not consecutive steps the multimedia system performs one after another; in any one streaming application, generation, transmission and presentation would normally be expected to occur concurrently, and continuously.

Instead, it represents the process the media itself goes through and allows us to examine at what point the various functions within a distributed multimedia system have effect over the media, and what those effects are.

A simplified analogy might be to a pipeline network, with valves at multiple ingress and egress, and flow controls at midway points. Instead of considering the parts of the system an arbitrary sample of content takes as it flows through the pipeline, we are studying the sections that make up the system: the pipes and valves at the input, output, and within the network; where and how we feed content into the pipeline and deliver content from it; what is needed to build and run these sections, and the control needed to manage flow between them.

3.1.2 Functionality within the lifecycle

At each stage in the lifecycle, the technologies introduced in chapter 2 are used to enable its operation; some are used in particular stages, others are used throughout. Using a coarse categorisation, the elements that provide this functionality can be assigned to layers within the system:

1. Transport.
2. Media.
3. Metadata.

The *transport* layer includes the network and interfaces to network services. Where group communication is required, the mechanism to provide it (multicast) is part of the transport layer. It also includes network quality of service provision, and protocols to overcome any deficiencies in the network not handled by QoS. The transport layer is core to the transmission stage of the lifecycle for streaming media, though its functionality must still be exposed in the generation and presentation stages to enable informed operation of higher layers.

The *media* layer builds upon the transport layer with functionality more specific to the multimedia application at hand. The multimedia content will be encoded using a format suitable for the underlying transport (e.g. data-loss tolerant, amenable to down-sampling) during generation, and correspondingly decoded (e.g. ensuring correct data ordering) at presentation. Control of the flow of the stream also falls within the media layer, encompassing any protocols used to start, stop, and navigate within the stream.

The *metadata* layer is the most intricate, and incorporates structured information related to the streaming media. In this definition, metadata is being used as a broad term to encompass the many forms of data that can be used to structure and annotate a piece of streamed media. A traditional hyperstructure can be thought of as metadata associated with a set of documents [78]; it should

also be expected to associate hyperstructure with multimedia documents that include streamed temporal media [30]. This metadata might describe the media (e.g. type, format, length, resolution), or the subject matter encapsulated within the media. It might apply to the media stream as a whole, or only to a segment of time within it. The metadata might be used to inform display of the multimedia through a user interface, or express relationships within the media or to other media, documents or metadata.

As discussed in section 2.4.2, structured information can also be derived from, and applied to, multimedia data in the media layer: parameters regarding the encoding medium, means of transmission through the network transport layer, and control of the streaming media. This (specific) structured information from the media layer can be bridged into the (generic) metadata layer where it can be used inform the processing of metadata not directly derived from the streaming media, e.g. for synchronisation with media for presentation. Similarly metadata of this sort might be used to adaptively control the transmission of streaming media. This media-derived structured information is considered metadata from the point at which it can be handled and processed separately from the multimedia stream.

3.1.3 Technologies in the Lifecycle

Figure 3.1 shows the transport, media, and metadata layers through the three stages in the media lifecycle. Example technologies which can provide the necessary functionality within a distributed multimedia system, and which have been described in chapter 2, are mapped within this space.

To illustrate how these technologies can be used through the lifecycle, the following simple example considers each of the three layers. In this example, a video clip is stored on a server for streaming; the video is a recording of a performance of a play:

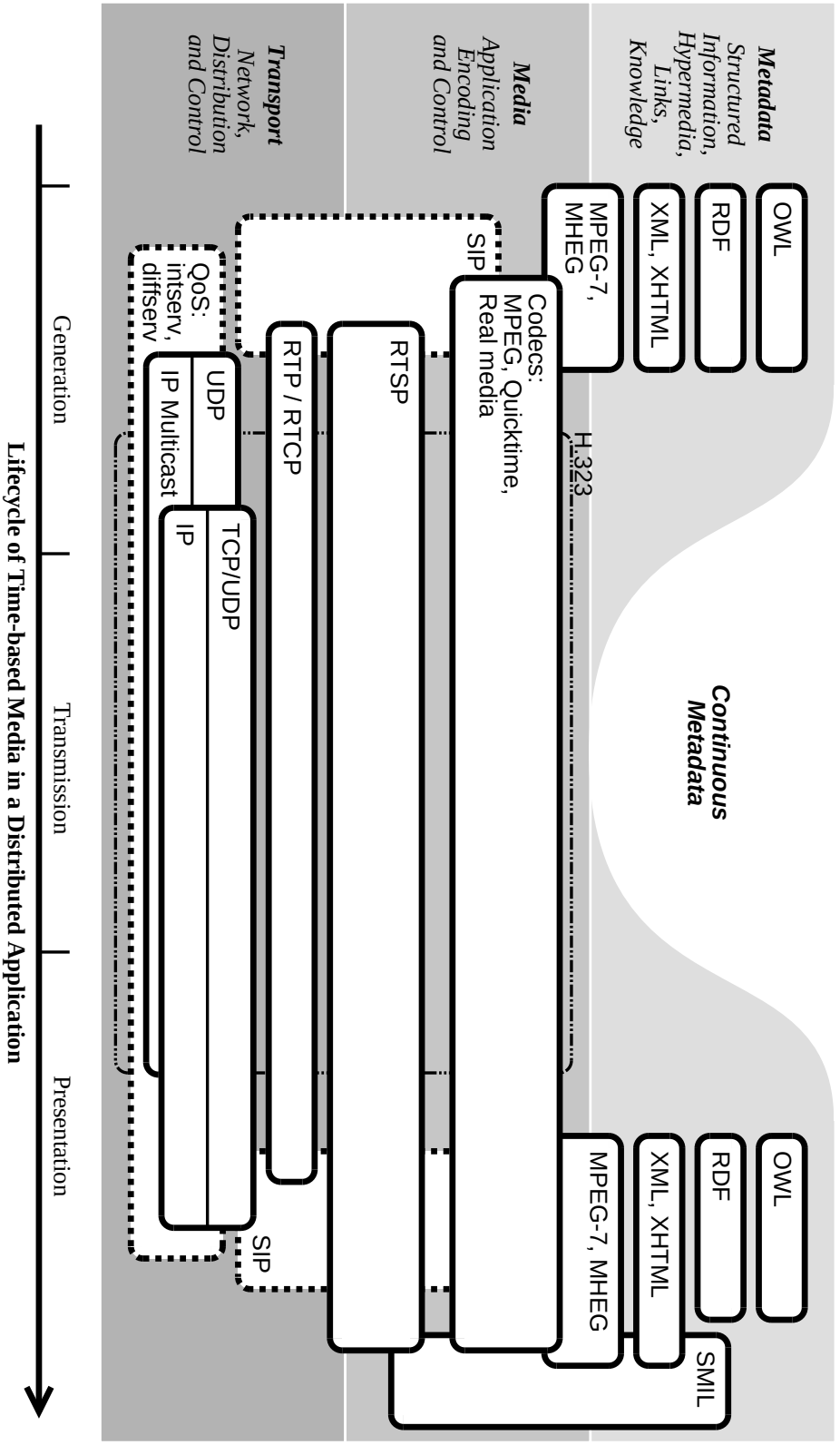

FIGURE 3.1: The lifecycle of streamed multimedia

3.1.3.1 Transport

In the transport layer, the encoded media is transmitted over an IP based network using RTP/UDP. Although the codec used by the media layer can overcome some packet loss, enough network bandwidth must be available to carry the majority of the data; this could be achieved by using IP quality of service reservation or prioritising. RTCP is used to monitor the RTP transmission, and report problems if they occur to the media layer (e.g. a reduction in available bandwith, the rise of a network bottleneck).

3.1.3.2 Media

In the media layer, the video (and some associated metadata) would be manipulated into a suitable format for streaming, such as MPEG-4, utilising a conforming media codec for efficient transmission of the video (such as the QuickTime MPEG-4 codec).

Control over playback of the video, including start, stop, pause and jump-to instructions, are communicated from the user to the media server using RTSP. Changes or restrictions in quality of service passed up from the transport layer might lead to renegotiations in the encoding of the media (e.g. bit-rate and resolution).

3.1.3.3 Metadata

In the metadata layer, RDF can be used to express a "seeAlso" relationship with the document containing the text of the play, and the Dublin Core vocabulary (also expressed using RDF) to describe attributes such as the title and creator of the recording. More advanced RDF could use an OWL ontology to express the relationships between the various forms of the play (i.e. the script and recording). This metadata could be held on the same server as the media file, and could be published, along with a URI for the video, via a web page.

Descriptions of the storage features of the content (such as the format and encoding) and structural information about this particular recording (such as scene cuts and other production directions) could be captured using MPEG-7, and transmitted along with the video in an MPEG-4 stream.

On presentation to a user, relevant information (e.g. the script, other works by the same author) are retrieved using the metadata. The visual and temporal layout of the video and associated documents, along with navigation between the elements, could be described using a language such as SMIL, which could also be used in conjunction with XHTML or other hypertext tools to link to the further relevant information.

3.2 Extending Metadata Throughout the Lifecycle

By categorising the functionality of a distributed multimedia system into transport, media, and metadata, layers, and mapping the extent of these layers through the continuous media lifecycle (figure 3.1), it becomes apparent that while transport and media layer functionality are present at all stages of the lifecycle, metadata use and functionality is concentrated in the generation and presentation stages.

While techniques exist for annotating structure for a piece of temporal media during generation, and utilising and displaying this metadata at presentation, there is no means by which to process and manipulate associated metadata during transmission, despite transmission being the defining feature of streamed multimedia.

This is not least because metadata tends to be applied when considering temporal media as a finite, closed, encapsulated, element within a document, rather than considering structure within the stream itself - as the streaming takes place. When referencing occurs within time-based media (as defined by the

Amsterdam Hypermedia Model, and implemented in SMIL), it takes place in the context of a presentation, where the streaming media forms an embedded element of a multimedia document, albeit a temporal one.

When annotating continuous media, time should not be limited to serving as an index into what is otherwise an isolated and self-contained media object. Just as the media changes and evolves over time, so does the metadata associated with it. The same principles of structured data, hypertext, and knowledge, apply continuously to the content represented by a media stream as it progresses; the hyperstructure itself has a temporal dimension, and must be represented throughout the lifecycle in parallel with the media.

3.2.1 Timeliness of the Transmission Stage

To see where and how metadata can be used in support of temporal media, we first categorise the multimedia data on two axes: *when* it is transmitted, and *how* it is transmitted. For each of these we consider the implications for the associated metadata.

Stored multimedia data, such as audio or video recordings, are persistent entities which can be described by their associated metadata in exactly the same way as any document, but with a temporal element. In a media-on-demand context, the metadata might be used to assist in finding, delivering and navigating the multimedia material; for example, the creation of movies by assembling video clips ('sharable video', see [112]) requires and creates metadata. The quantity of meta-information is likely to be small in comparison to the size of the original material. There may be little justification for streaming the metadata, which can be processed in advance of streaming the multimedia information, unless the metadata must be streamed due to sheer volume of data.

With *live* media, the metadata describing the structure of the content might not be available in advance, instead becoming available during the generation of the multimedia stream (or streams). For example, this metadata could include

information about camera positions, or decisions the producer is making on-the-fly; live interactions might generate links [116]. It could also result from real-time processing of the stream, such as some form of classification, segmentation or annotation.

In some cases although the multimedia content is *stored*, it will be viewed as part of an interactive (or collaborative) user experience. Any metadata generated through live interactions with the users as it is *viewed* (rather than recorded or stored) will be *live*, as if the media were too.

It is sometimes acceptable to introduce a delay in live media, which can give time for a pipeline of intermediate processes. An analogous situation can arise if the multimedia data takes a long time to present: the first viewer of a presentation lasting several hours may provide useful annotations for a second viewer who accesses the presentation before the first has completed authoring - if metadata were preloaded at the start of the presentation the second user would not benefit from the information provided by the first (although the media stream is not live, the metadata is). In both cases preparation of the metadata cannot be guaranteed in advance, and must be streamed at the same time as the multimedia content. In some other situations the streaming of pre-existing metadata could be necessitated by the absence of any other form of metadata transport; for example, a receive-only device joining a live radio or TV broadcast at an arbitrary point in time.

Live media that connects two or more parties in *real-time*, such as video-conferencing, is the most demanding scenario. Session and group membership metadata may be available in advance, but content metadata is created on-the-fly and there is little opportunity for any pre-processing, as there are tight time constraints on this style of synchronous interaction. For example, the anchor generation system in OvalTine [134] was designed with video-conferencing in mind. Collaborative virtual environments also impose real-time aspects, together with the prospect of a wealth of metadata associated with the objects involved as well as the people.

3.3 Continuous Metadata

It is apparent that, especially in live and near-live scenarios, there are compelling reasons to stream knowledge and structure, captured through metadata, in parallel with the parent media stream. In doing so, the metadata layer is extended across the entire temporal multimedia lifecycle, from generation through to presentation, bridging the gap in the transmission stage (figure 3.1).

Although the metadata referred to is streamed, it is preferable to use the term *continuous metadata*. The word 'stream' has become closely associated with real-time audio and video, and often (incorrectly) implies a non-stop flow of relatively high bandwidth data. Streaming is a *mechanism* for transferring multimedia data; metadata might not use the same mechanism, and whatever mechanism is used it is of less importance than the continuousness of the metadata.

While continuous metadata is transferred in a stream-like way through the transmission stage, this does not mean that the same processes and techniques as applied to streaming media data are suitable. This section considers the facets of the metadata layer which need to be carried across the transmission stage in the context of chapter 2; chapter 4 compares and contrasts the requirements of continuous metadata with those for the media and transport layers during transmission.

3.3.1 Generalisation of Structure

As seen in chapter 2, structure is available and used in several different ways and forms. In many ways, metadata can be seen as adding structure to information: a hyperstructure can be considered metadata to a multimedia document; similarly, within knowledge systems, metadata can be used as the structure by which information is interpreted into knowledge, through reasoning and inference. Metadata in itself does not presume structure, but since the primary

payload of continuous metadata is likely to fabricate structure, the elements of this that are taken through the transmission stage should be considered.

Conklin described links as the structural core of hypermedia systems [45]. Links are the manifestation of structure in hypermedia systems, and linking structure should be one of the forms supported by continuous metadata. Additionally, the five defining features of an OHS [52] are entirely compatible with the notion of continuous metadata:

1. *The system does not impose any markup upon the data*: continuous metadata annotates media streams, it does not modify them.

2. *The system can integrate with third party tools*: continuous metadata can be used by tools at generation and presentation in a similar way to other metadata, but with added temporal elements.

3. *The system can be distributed across networks and hardware platforms*: continuous metadata is explicitly conceived to support distributed multimedia.

4. *The system does not distinguish between readers and authors*: there is no discrimination as to where and when the generation stage takes place (mixed flows of authoring and presenting continuous metadata are further explored in chapter 4).

5. *The system should allow the easy addition of extra functionality*: continuous metadata does not dictate the type nor format of any metadata; enhanced functionality is empowered through new classes of metadata and structure.

This should not imply that continuous metadata is primarily conceived as an OHS, although it maintains these ideals. Instead, it supports hypertext systems - specifically links - as one of many types of metadata, part of a wider distribution of structure [109].

The commonalities of structure as metadata bring together the fields of hypertext and knowledge, as can be seen in semantic web activities. For example, a directional link can also be represented as an RDF triple:

- the *subject* is the source endpoint / anchor.
- the *predicate* connects the subject and the object - it is the link,
- the *object* is the destination endpoint / anchor.

Continuous metadata should be a mechanism for bringing all these types of structure - of metadata - to bear on the application of distributed multimedia.

3.3.2 Separation of Structure

Continuous metadata is fundamentally annotation of a streaming media resource. It is, however, a flow of data, and consideration must also be given as to how the metadata manifests itself in relation to the resource it augments.

The open hypermedia community has long advocated the separation of hyperstructure from documents [51] to enable the use of alternative and multiple views annotated by different, possibly distributed, linkbases; essentially, allowing the hyperstructure to be processed independently from the media it relates to. Despite earlier use of embedded links and structure in HTML [118], these ideas are now being explored by the WWW community through Xlink and XPath [58, 44]. Semantic Web activities also support this model through the use of URI scoped resources, described by separable namespaces and structure definitions (which are also resources) in standards such as RDF [94] and OWL [18].

In many ways, this approach is at odds with the manner in which markup is used in most existing streaming media systems. While the evolution of multimedia technologies and standards now promotes the capture of metadata 'upstream' in the production process (e.g. shots, script, storyboard), it is normally embedded within the media stream during transmission. The MPEG standards are evolving

(through MPEG-7 [103]) to accommodate this, and associated metadata, within a combined data stream (MPEG-4 objects). While there are advantages to transmitting and storing multimedia data with the metadata embedded in this way, there are also situations where metadata should be handled separately and delivered synchronously.

In particular, the separation of continuous metadata allows for a much more flexible framework of distributed delivery, processing and presentation. Thus far, consideration has primarily been of a single continuous metadata flow associated with an individual media data stream. Many more possibilities become apparent when there might be several media streams, and more importantly, many independent and distributed continuous metadata flows.

Open Hypermedia Systems derive a great deal of their flexibility from the ability to apply different hyperstructures to the same document, using different (possibly distributed) linkbases. Knowledge can be sourced from many places; many structure services can be called upon the information. The WWW achieves its flexibility through the wide distribution of (more simply structured) information.

These ideas can also be utilised when considering continuous metadata. By separating the metadata flow from the media stream, the sources of metadata can be widely distributed, and independent from the media provider. By using standardised references into the media, these different metadata sources could provide a user with different perspectives and structures (in effect, linkbases). Being flows of knowledge and data in their own right, the continuous metadata could be processed independently at other distributed locations to tailor the final content to the requirements of the user, merging, processing and adapting several flows together into a continuous chain of knowledge. An individual user might elect to receive only a handful of metadata flows from a choice of many; these flows, along with the media stream, could then be presented onto the user's viewing device (perhaps using a temporal layout markup, e.g. SMIL).

Separation of continuous metadata also supports situations where the media stream reaches a user through a different medium. For instance, while an

audio-video stream might reach the user through a traditional television broadcast (e.g. cable or satellite) the metadata may arrive through an Internet connection.

3.3.3 Temporal Hypermedia

Temporal hypermedia models (such as the AHM) and standards (such as SMIL) are primarily used to define and structure presentational semantics. The metadata they utilise is mostly confined to the presentation stage of the lifecycle, and is not directly relevant to the transmission stage and thereby continuous metadata. Continuous metadata would more likely provide a further source of information, changing over time, that would be incorporated into a temporal hypermedia application, and synchronised and laid out using a SMIL or similar.

3.3.4 CSCW

Continuous metadata is suitable for supporting all four classes of CSCW applications: messaging systems; computer based conferencing; meeting rooms; and co-authoring and argumentation (all these are present in the motivational scenarios in section 3.4). It also is applicable throughout the CSCW interaction/location classification, being useful for both synchronous and asynchronous interaction.

3.4 Motivational Scenarios

Having presented an exposition of the concepts behind continuous metadata, the following section introduces several motivational scenarios in which these ideas could be employed.

3.4.1 Live News Broadcast

Broadcasters can currently classify programmes in their schedule using more traditional metadata. By maintaining records including the time of a programme, its name, presenters and a brief overview, viewers can access this data through one-way mediums encoded within the analogue television signal, such as teletext. More recent Digital Video Broadcasting (DVB [62]) can present this information within an Electronic Programme Guide (EPG), presenting the user with "now and next" information. As and when schedules are changed, this data can be pushed to the user over normal broadcast channels.

With the advent of digital personal video recorders (PVRs), random access storage of media is available into the home, and broadcasters are beginning to implement simple content classification systems in addition to schedule reporting [61]. This allows the recorder to track the types of programme a user watches, and infer that it should speculatively record other programs of a similar type, through use of an agent based recommender system [47, 50]. The classification of programmes is often performed by hand, limiting the amount of markup applied, although coarse-grained automation has been developed [143].

In these current systems, classification and markup must be performed before the programme is broadcast, and applied at a granularity which is the length of a programme; no knowledge is transferred about sections of material within the programme.

Through use of continuous metadata, information of relevance within, and throughout, the programme can be sent to the user. This is particularly true in the case of live programmes, when although there might be extra information available, the exact detail of content within the programme is only known a short time in advance. Rolling news channels are a good example of this.

Using continuous metadata flows that are separate from the video stream, the main video could be sent to the user with traditional broadcasting techniques. The metadata flow could be delivered using a lower bandwidth internet

connection (cable modem, xDSL etc.) to the home, and then to a suitable display device within the home. This might be integrated into a television, or more likely a PC, or wireless handheld device. Where aerial or satellite reception were not available, for example programmes from international or minority broadcasters, the video stream might also be sent using internet protocols; one user might watch programmes mixed from several broadcasters received in a multi-modal manner.

Any programme content classification could now be generated dynamically as the programme was filmed live, and transmitted on a separate continuous metadata flow. For example, a user might choose to receive the "programme description" continuous metadata flows from multiple broadcasters while only watching a single media data stream, which is constrained by bandwidth (for Internet streams) or equipment limitations (for traditional television receivers). The display device could monitor the content metadata for all programmes currently being broadcast, and switch the video data to prioritise a channel broadcasting, for example, sports bulletins.

Links to relevant web or hypermedia documents also constitute metadata, and can be delivered as a continuous metadata flows [110]. The broadcaster of live television news might provide a flow of links to documents on their website, corresponding to the current news item, or to interactive discussion boards and messages within them. Following a link might pause the temporal display of video stream, recording it to a PVR, for time-shifted playback after reading the document.

Links are not limited to referencing non-temporal documents, but could also begin streamed playback of archived video footage, programmes or reports. These in turn would have their own archived continuous metadata flows, in addition to those running live.

Distribution of continuous metadata sources would enable the broadcaster to franchise their media stream, allowing other news commentators to generate link flows containing their own links and comment. A user could select different

suppliers of augmentation according to their own taste and preference, rather than that of the television company. Communities of like-minded viewers might even form to share insight, using webs of personal metadata flows.

A user might also run a recommender system locally, to analyse link trails previously followed, and select the best suited links from the many incoming metadata flows (or perhaps create a new link flow source based on that users' browsing history).

Subtitles, which are normally raster images superimposed over a video stream, are essentially a text augmentation, which could also be separated into continuous metadata. The television broadcaster might provide a subtitle flow in the native language of the programme, typed live along with the broadcast (and with advances in technology, perhaps automatically generated or assisted).

Using distributed continuous metadata sources, translators could provide flows of subtitles in other languages, independently from the original broadcaster. One agency might provide translations for several television channels, using the same infrastructure. This could be performed manually, with translators using the original video stream; or, since the primary subtitle flow is already text based, basic translation could be automated.

3.4.2 Musical Performance

When performed, music is often heard as a linear progression of sound; when recordings or live performances are streamed, it is unsurprisingly done so as a simple progression of data encoding that sound. In fact, music is rich in structure and has many parallels with more traditional hypertexts [53], as a musical score clearly shows.

Music, by its very nature, contains large amounts of structure at various levels. The basic notes and performance directions (tempo, volume, style) a composer uses to capture a piece can all be used to describe that which we normally work

with in the audio domain. In turn these can be used to derive musical structure. Many works exhibit a musical form (binary, ternary, variation, sonata, and suchlike these are terms established in the western music tradition) and within these employ structured repeats and codas. Melodic themes are exposed, modified, and re-introduced, while chords often follow well-defined progressions.

While this structure can be annotated on a musical score, other representations of the same content (for example, a recording) do not explicitly carry this metadata. Indeed the same music can be available in several different forms, each containing different types of structure: digital audio, MIDI, Standard Music Description Language (SMDL, a HyTime language [91]), scored representations, and so on. Some composers and musicologists use techniques such as Schenkerian analysis [135] to attempt to derive further higher level structure from music, which might be based on pitch, temporal elements, or both.

Continuous metadata flows could be used to enhance the experience of performers and listeners through use of these structures, and utilising the interchange between them.

While listening to a piece of recorded audio, streamed from a remote digital library, a user might also have access to a digitised copy of the composers original score, held at a different remote library. As the user listens to the music, the first library is sending a corresponding continuous metadata flow of spatial co-ordinates, referencing an area within the digitised image of the score held by the second library. As the music plays, the score moves in time on the users display. Accompanying programme notes might also be included as a metadata flow of hyperlinks; the user might also make their own annotations and links to share with colleagues at a later point.

Should the user wish to pause or navigate temporally using the audio media player, the metadata flow will be adjusted accordingly, and the score image updated to centre on the correct bar. Conversely, the user can select an area of the score to restart playback from, and the audio stream and metadata will revert to that point.

Since the metadata flow is continuous, similar tools can be used when listening to a live broadcast of the same composition, including score display. The user might even want to compare notes from listening to the recorded library performance against the current live one.

Continuous metadata could also be used by the performing musician. While rehearsing, a musician might transmit a stream of their music to a specialist node which transcribes the audio into a MIDI metadata flow (a MIDI enabled instrument could provide the metadata at source).

This metadata flow could then be sent to a remote music repository, where the melody would be analysed for a match in its database [23]. Using a combination of the match from the database and the original MIDI metadata flow for tempo, the library could output a new continuous metadata flow of MIDI data containing an accompaniment for the melody, which the rehearsing musician could receive and play along to. (There is also obvious entertainment value in playing along with recordings of musical heroes in a similar manner!).

Music-making is often a collaborative process, and sharing structure with continuous metadata brings possibilities for distributed rehearsal.

3.4.3 Lecture Presentation and Laboratory Experiments

A lecture presents an interesting environment for capturing and using media and associated metadata (figure 3.2). Different media types are likely to be used, and while the lecture itself is clearly live, remote attendance and revision are likely to require some method of playback.

The lecture could be recorded digitally to audio and video formats throughout. This is the most detail-rich transcription of the lecture, and is both stored for later playback, and also streamed live so students can take the class at home if they wish. Some students who are late arriving may opt to receive the first part of the presentation as an audio only stream via the campus wireless network while they travel to the lecture theatre.

The first section of the presentation consists of both the spoken lecture, and a number of overhead slides. The slides are made available via the web, and as the lecturer transitions between them, references to the current slides are transferred over a continuous metadata flow to the students in the lecture theatre and at home, where their own personal display devices also update to show the correct slide.

When the lecturer makes annotations to the notes, or offers explanatory notes on the smart white-board, these are also sent via continuous metadata. The lecturer can use continuous metadata to cross-reference and replay foundation topics from previously recorded presentations. Students might also make their own notes, and should a friend miss the lecture they might play back the lecture at a later point, referencing their colleagues notes from a metadata flow.

All of the metadata flows can be recorded for future use, so students can revise the lecture at a later point. Navigation of the material at this time is likely to be a less linear affair; temporal positioning might be through VCR-like control of the video stream, selecting a particular slide and replaying the video explanation, or from a students' own notes. Late students, listening to audio only, might place brief markers during points they don't understand, which could be used to revisit these points after the lecture, or when they arrive, using the full range of video and slide material.

A lecturer may have recorded a class so students can later revise that presentation. The lecture itself would be replayed using audio and video mediadata streams, but in addition the author could provide links to the relevant points in overhead slides or online notes as a metadata stream. The presentation point would then display the parts of the notes to coincide with that temporal space in the lecture.

In the second half of the lecture, a demonstration is made of practical work to be undertaken by the students in a later laboratory session. The experiment involves performing a series of steps to induce a chemical reaction between two materials. Audio and video streams are still collected, with a further video

stream now focused on the experiment to capture the visual elements of the reaction (colour, bubbling gas etc.); slide transitions continue as before. In addition, various parameters from the experiment, such as temperature and acidity, are measured electronically, and the results are stored, and delivered, with continuous metadata.

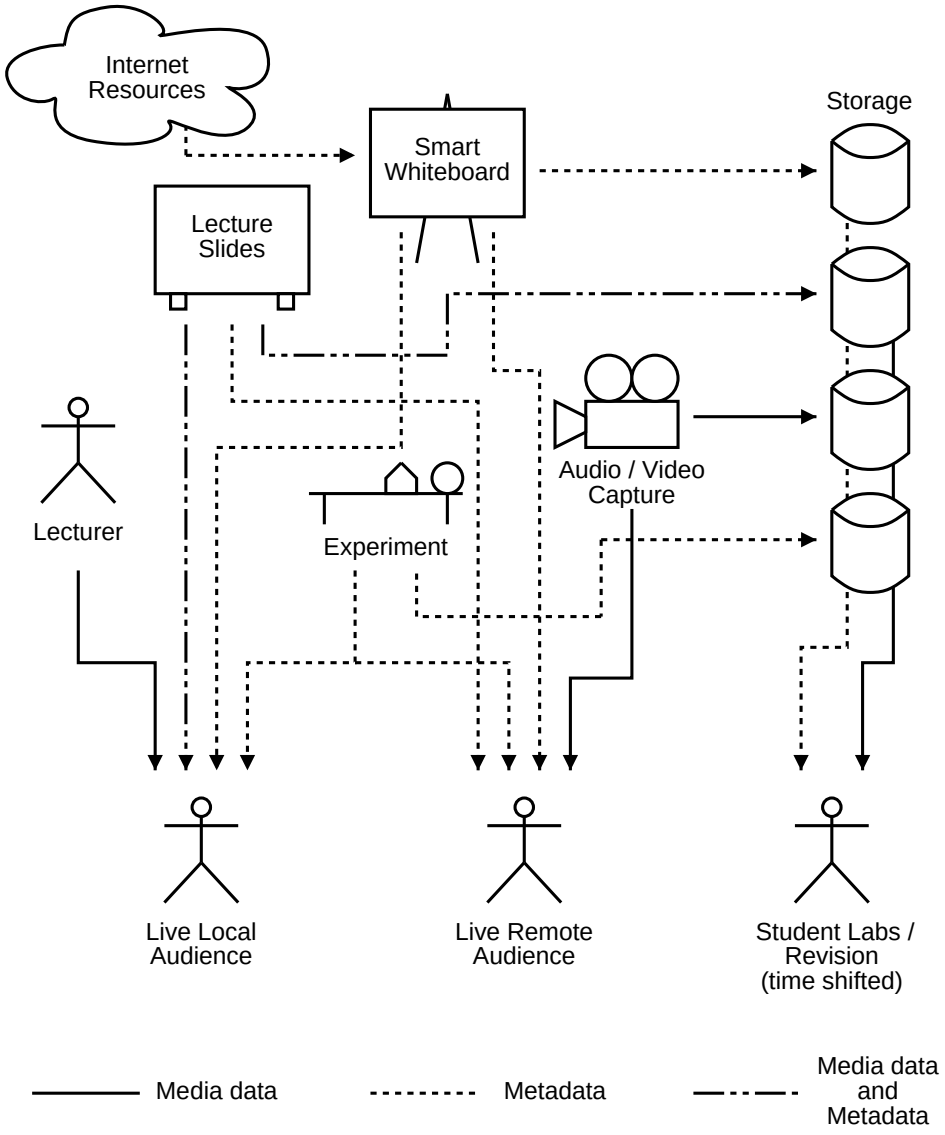

FIGURE 3.2: Some of the information flows in the lecture and laboratory scenario

At a later point, when the students undertake the laboratory experiment, their equipment is set up in a similar way to that of the lecturer, with measurement devices attached. The streamed recording of the lecture is available, along with the presentation slides, and any annotations from the lecturer and students.

As the experiment progresses, students can navigate the recording of the demonstrations, navigating within the media as necessary, and compare their own results with those of the lecturer during the demonstration.

The measurements from each students' laboratory bench are captured by individual continuous metadata flows, both for later investigation (an automated lab book, which the student might annotate), and so that supervisors and lecturers can track progress.

The results metadata might be fed into graphing and other visualisation tools, so that a student can compare their progress to the lecturers demonstration and their fellow students. Since the result metadata flows are also temporal, any slides, notes and annotations from the lecture can be navigated to from the results data; as the results progress, relevant sections of the lecture will be updated.

The lecturer might also suggest that students utilise a specialist knowledge base for that particular subject area. The knowledge base would interact with the framework though a filter node, receiving the metadata flow of the lecturer's notes and transmitting a further metadata flow of links based on its processing of the original notes.

Closer to the presentation point, a student may have a further knowledge base of personal preferences, built up from their previous browsing history. This too would interact through a filter node, adding (or removing!) links to external information tailored to that individual.

An enterprising student might then wish to share their personal knowledge base with the rest of the class, a mechanism the student could also use to distribute any insights or annotations they have added to the lecture.

3.4.4 Collaborative Distributed Meeting Spaces

Distributed meetings are a further scenario with rich media and metadata sources that can benefit from continuous metadata [13, 17]. Here, a meeting or seminar might take place across several physical locations, with the participants joined through a conferencing systems (e.g. telephone-conferencing, or Internet video-conferencing. Recordings of the conference audio and/or video form the detail-rich resource for the meeting; by capturing metadata from other sources and tools in the rooms, a record of the meeting can be made available retrospectively as a structure-rich hypermedia resource.

Sources of metadata include:

- the meeting agenda, which sets out the topics to be discussed.
- a group memory capture tool, with which a scribe takes minutes in the form of a light-weight hyperstructure.
- a tool to manage structured tasks, through which people are assigned jobs. These might be reviewed at a future meeting, or at points in between meetings.
- gated microphone headsets, which can be used to record who is speaking at a point in time.
- logs from an instant messaging text chat-room the meeting participants can use for "back-chat". This is a resource in itself, although metadata assertions can be made when a person commits a comment.
  - in virtual meetings with international participants the text channel could be used by interpreters to provide a translation of verbal discussions, similar to subtitles in the live broadcast scenario.

As continuous metadata is produced, it is distributed to the other participating sites in the meeting in tandem with the media stream. Here it is used to enhance

the video-conferencing experience, for example highlighting the name of the person who is speaking (which is not always clear from the video picture).

As the metadata is stored, it becomes part of the meeting hyperstructure. Because there are several separate sources of metadata, an ontology is used to mediate between the different vocabularies. These sources of metadata, although primitive, are cheap in resource terms compared to marking up audio/video by other more intensive means - the goal is to use as many sources of structure as possible that might be available in an enhanced meeting environment, and combine them to provide more powerful knowledge than they contain individually. The hyperstructure for the meeting can then be used to navigate into the detail rich-media recordings, both within the current meeting e.g. to go back on the detail of an earlier point, or in following related meetings e.g. to provide the agreed minutes in an agenda.

This scenario, in particular, is explored through practical experiences in chapter 5.

Chapter 4

A Conceptual Framework to Enable Processing and Delivery of Continuous Metadata

In the previous chapter continuous metadata was introduced to support the metadata layer throughout the transmission stage of the distributed multimedia lifecycle (figure 3.1). The aspects of the metadata layer that continuous metadata should support were then described.

The first section of this chapter examines the other dimension of the lifecycle, comparing and contrasting the properties of continuous metadata with those of the media and transport layers, particularly during the transmission stage.

The second section of this chapter applies these findings in the description of a conceptual framework for distributing, integrating, and delivering, the metadata which can accompany temporal multimedia data in a distributed system. It does not specifically deal with the 'content' of the metadata, instead focusing on the importance of how the metadata is sourced in the stages leading up to presentation, and breaking the functionality down into a set of abstract components.

4.1 Comparison of Continuous Metadata and Continuous Media Properties

This section revisits areas introduced in chapter 2 which impact upon the transmission of multimedia streams in the transport and media layers, and compares them with the properties required of continuous metadata. Although continuous metadata has an analagous role to streaming media in the transmission stage, differing characteristics require alternative mechanisms to support it.

While essential for live applications, streaming media is often used to overcome the significant delay needed to pre-load sizable amounts of multimedia over a network with comparatively small throughput. Many of the techniques introduced to support continuous media have been developed to accomodate the fundamental need for it to be streamed in a timely manner, primarily because of the volume of media data in comparison to the capacity of the network transport. Associated metadata would normally be expected to form a much smaller quantity of data, and so the reasons for streaming are different (section 3.2.1). For shorter volumes of media it can be argued that the metadata can be pre-loaded into the client by downloading one file before presentation of the media begins. For greater lengths of media it might be the case that the amount of metadata has become large enough to warrant streaming, but it is for live and near-live broadcast of media that streamed metadata becomes essential.

Since the primary factor when dealing with streaming media is the quanitity of data in comparison to network throughput, many of the transmission techniques are employed in the media as well as the transport layer. An acceptable compromise is made to reduce the quality of the media using lossy encodings, to balance against the limited bandwidth and QoS guarantees provided by the network. In this way the *type* of the media is often modified to accomodate streaming. In comparison, the type and encoding of continuous metadata has less dependance on the capabilities of the underlying infrastructure, because the

metadata places a lighter load on the network. In the framework introduced later in this chapter, it is not the type, encoding, nor content of the metadata that is important, rather that it is some kind of metadata and that it is handled in a continuous manner. As it is the structure of the metadata that is of primary importance, the encoding is better tailored at a higher level, to the domain of the application.

4.1.1 Real-time Distributed Multimedia

Continuous metadata can contribute towards *presentational* and *conversational* multimedia applications, and should therefore be considered part of a combined application [80]. While presentational applications only use stored or live media, and therefore metadata, conversational and combined applications require live *real-time* flows; indeed this is where continuous metadata is of most use (section 3.2.1).

Chapter 2 introduced the four resources [5] which are required by systems to support distributed multimedia applications, focussing on the requirements of continuous media. These can also be considered in the context of continuous metadata, and form the basis for discussion in this section:

Explicit Support for Continuous Media - Continuous metadata is conceived to explicitly support continuous media, and in this way can be seen as a contributing element to a distributed multimedia application. Continuous metadata has requirements of support and resources from the underlying system - as with continuous media - for the duration of the media at the presentation rate required, though some of these are independent and contrasting in comparison to those for continuous media.

Quality Of Service - Unlike continuous media, continuous metadata might not use a large amount of system resource for the duration of its presentation; its quality of service requirements are therefore likely to differ, although they still exist.

Synchronisation - Within a multimedia application, continuous metadata is a constituent flow of information which must be synchronised with other components for presentation. There may be further synchronisation requirements *between* continuous metadata flows within the transmission stage.

Group Communication - Continuous metadata incorporates the requirement to provide information to and communicate with groups of collaborating users, rather than one user at a time.

4.1.2 Quality of Service

Quality of Service mechanisms are required by multimedia streams because they usually have high requirements for network resources (especially with regard to bandwidth and latency) for the duration of their presentation. Network quality of service frameworks enable an application to request a level of priority or guarantee from the transport layer, and if necessary, adapt the media encoding at the media layer accordingly.

Since the volume of continuous metadata is lower than that for multimedia, best-effort delivery is likely to be sufficient when the transport is provided over a fixed network. Low bandwidth networks might necessitate a quality of service framework on grounds of capacity, but in most cases the *priority* of continuous metadata is more important. The transmission timing of the metadata has significance, and it will often be augmenting continuous, streamed, media data, with which is contending for network resources. Thus a balance must be made between providing enough bandwidth to deliver media with as high a quality as available, while still ensuring the timely delivery of continuous metadata; since this is essentially a prioritisation problem, class based mechanisms such as diffserv are likely to be sufficient, without the need for heavy-weight reservation frameworks such as intserv.

Two of the quality of service parameters introduced in chapter 2 [42] can be

considered differently in the context of continous metadata:

Media parameters - quantitativly, live metadata has different features in comparison with live audio, or still video, in particular with regard to latency and bandwidth, as elaborated in this section. Qualitative parameters are less variable, since the metadata either exists, or does not - characteristics such as resolution are not applicable.

Fault tolerance - continous metadata demands a high reliability and timely service; it is less tolerant of faults which could corrupt the metadata, although in some circumstances it may be possible to loose an isolated assertion without damaging the entire information structure (section 4.1.5). This is particularly important when considering parameters from the subjective viewpoint of a user, where the impact of incomplete or defective metadata might be more noticeable than a missing sub-frame of video.

As with streaming media, quality of service needs to be an end-to-end provision. Since the volume of data is less of an issue, and timeliness of greater importance, the QoS requirements for continuous metadata will be less consistent from application to application, and will change depending on whether the scenario relies on stored, live, near-live, or interactive elements.

4.1.3 Group Communication

Group communication is required for distributed multimedia, where the most suitable mechamisn for broadcasting streaming media is network multicast. Multicast is not required for continuous metadata, although it is highly desireable.

When used with streaming media, multicast reduces congestion on segments of the network which would otherwise carry duplicate streams. Since the bandwith requirements of continuous metadata are lower, these effects are reduced, however multicast does still help the scalability of metadata in line with media

flows, and removes the need for complex redistribution nodes to handle the many point-to-point flows that would otherwise be required.

Multicast is built upon the unreliable UDP protocol, which simplifies scalability of control and re-transmissions. While this is acceptable for streaming media, which use specialised encodings to withstand dropped packets, continuous metadata is more severly affected by data loss, and thus requires reliable multicast protocols (section 2.2.3).

4.1.4 Transport Protocols

Transport protocols for both streaming media and continuous metadata must be designed with an explicit recognition of real-time data paths. Protocols for streaming media have also been architected with high volume multimedia specifically in mind. RTP, commonly used for Internet delivery, is a straightforward UDP based protocol that keeps latency and delivery overheads low for the large amounts of data transferred, and is ideally suited to use with multicast networking.

In comparison, continuous metadata need not be high volume, and there may be significant lulls between bursts of data (although the transporting connection is conceptually open). Timeliness of delivery is critically important, but not in exactly the manner as for multimedia. While there is a smaller amount and flow of data, it has higher informational value, and as such tolerance of missing data is lower. Audio or video requires prompt delivery of packets - data which arrives too late is as good as missing, but can be endured by loss tolerant codecs - so the quality of the media feed is compromised to permit real-time display. The loss of a packet containing a constituent part of a metadata statement cannot be ignored as it will corrupt or invalidate that piece of information (unless each self-contrained statement of metadata were to match the size of a network packet[1]); if a packet is not delivered it must be retransmitted.

[1]It is conceivable that metadata statements of restricted size, e.g. an RDF triple with limited length URIs, could be self-contained within a single RTP/UTP packet. In this case, it may

In this way, continuous metadata requires a *reliable* transport protocol; the unreliable protocols used for streaming media are almost entirely unsuitable. In particular, to support group communication, a reliable multicast protocol is needed.

4.1.5 Content Encoding

Encoding of streaming media is strongly tied to its intended use, and the network resources available. Specialised codecs are developed to provide the best quality to compression trade off for different applications, e.g. for video or audio, and subtypes beyond this, e.g. spoken voice or classical music. Furthermore, the use of lossy codecs when encoding the media gives the stream some resistance to packet loss in the transport layer, since the codec can tolerate some missing data without compromising the entire stream. Similarly, codecs can allow down-sampling of the media so as to make a trade-off between ensuring delivery and the availability of bandwidth (section 2.2.5).

While the type of information carried in the continuous metadata is clearly significant on a per-application basis, where domain specific markup and ontologies will be used, from a more general viewpoint it is the transmission timing of the metadata which is paramount. Classification and exchange of metadata and structured knowledge can already be described by standards from MPEG-7 and Dublin Core through to OWL (section 2.4); continuous metadata leverages such vocabularies and ontologies, but does so in a continuous and temporally significant way.

The encoding of continuous metadata is therefore more strongly tied to the application domain, rather than the transport layer; any encoding presumes the metadata will be delivered intact, complete, and on time. Techniques for

be possible to continue building the information structure while suffering the loss of individual packets/statements if the structure is resilient to segmentation. The information structure would also be vunerable to inconsistencies across its distribution, where different nodes have received a different series of metadata statements at any one time.

encoding media streams are entirely unsuitable, since all the metadata is required and cannot be interpolated.

If encoding is required for compression or encryption, it should be selected based upon the markup of the metadata, which is likely to be text-based. This would essentially form a wrapper around the continuous metadata, without changing its intrinsic function. XML encryption processes may be suitable [89], and standard compression techniques such as GZIP [60] could be employed; other novel techniques exist for structural and semantic compression of XML [96], although they are performed on a complete document, so are less suitable for streamed data.

4.1.6 Signalling and Control

Distributed multimedia applications require mechanisms to co-ordinate and regulate the different components a particular tool might use, which are likely to include multiple channels of audio and/or video, and shared applications (e.g. distributed whiteboards, slide presentations). Solutions in use include RTSP for playback control , SIP for initiation, and the ITU H.323 family (which includes the T.120 protocol for data transfer channels) (sections 2.2.4, 2.2.6). Continuous metadata should not replicate or include the functionality of these frameworks, since it can be considered a component to be used within, and controlled by, them.

4.2 Metadata and Mediadata Flows in the Framework

The rest of this chapter takes the points raised when considering how to extend metadata through the transmission stage (section 3.3), and comparisons with streaming media (section 4.1), and generalises these into a conceptual framework to process and deliver metadata.

In this framework we consider mediadata and metadata, and define a *continuous metadata* flow as one which carries additional data about a corresponding temporal multimedia, or *mediadata*, flow. The mediadata will normally be a multimedia stream, such as audio or video, and can be characterised as a continually evolving flow of data, where one frame of a video generally has a direct relationship with the previous. Metadata, on the other hand, will be split into discrete chunks of information within the continuous metadata flow, where each is an assertion about the mediadata. While these assertions may well build upon each other to form part of a greater information structure, and assertions about the same resource may modulate over time, the metadata cannot be interpolated between in the same way as mediadata.

Some flows of metadata might be extracted from structured information in the mediadata, especially physical and medium-based information (section 2.4.2), but also perceptual and transcriptive annotations available in suitable encodings (e.g. MPEG-7 through MPEG-4). The classification of this information as metadata within the framework does not preclude its use in association with mediadata, for example to control streaming rates and playback, nor later recombination of the metadata (potentially enriched following processing in the framework) with a mediadata stream.

The metadata is transported through the framework separately from the mediadata, rather than multiplexed and carried within the same flow. This follows the open hypermedia convention of separation of links (where links are metadata to a document; see section 2.1.2), and so also allows for a much more flexible framework of distributed processing and presentation with different and distributed information sources and multiple transmission routes (discussed later in this chapter; also see figure 4.1).

Sometimes the distinction between mediadata and metadata is not as clear - it can be argued that what may be metadata in one case could be mediadata in another, and in many ways this is true. For example, a flow of MIDI information would be metadata for a raw audio mediadata flow in one case; in another there

may not be an audio stream and the MIDI might form the base mediadata flow which is then augmented by other metadata. The framework should be flexible enough to support both these cases, but clarification is required. Since we are working in a highly temporal system, we define the mediadata flow to be the one against which the timing of metadata flows are made; and in most cases it is desirable to designate the mediadata flow as that which carries the high volume multimedia information (since it will have the greatest volume and number of associated metadata flows).

We should also note that just because a metadata flow may develop a derivative metadata flow "about" or from it, this does not make the descendant flow "meta-meta"-data, nor does it imply the original metadata should become a mediadata flow. The derivate flow merely becomes another metadata flow based on the original mediadata, albeit one with a more complex relationship with other metadata.

4.3 An Initial Framework for Unicast Flows

We will first consider how the framework should handle point-to-point media and metadata flows by introducing the various elements which make up a simple version of the framework.

4.3.1 Sources and Flows

There must be a point at which the **mediadata** enters the framework, and we refer to this point as the mediadata *source*. For simplicity, we initially presume that each mediadata flow is injected from a single source; with a more complex implementation there is no reason why a mediadata flow cannot enter the framework in a distributed manner from several points arcoss the network (pehaps more locally to the intended recipient). The method by which the content of the mediadata is transported through the framework should be

suitable for that data type, e.g. RTP for audio or video (section 2.2.4). The framework should be relatively agnostic with regard to the method of transport, although the protocol used *should* ensure timely delivery and *must* be able to provide identity and timing information to the framework (for presentation and synchronisation purposes, so metadata can be asserted about the mediadata).

The **metadata** source is the point at which a continuous metadata flow enters the framework. This may be from the same physical location as the mediadata source or it may be distributed from a different point: e.g. for a live news feed a provider might construct a metadata flow of relevant links at the same broadcast point as the mediadata; when viewing a video of a pre-recorded lecture a user may wish to receive metadata annotations from a source other than that of the original lecture.

The continuous metadata source must always output information in a temporally relevant manner, and to do so it may require a flow of mediadata from the appropriate source from which to derive the metadata or obtain synchonisation waypoints. If only timing information is required this could itself be derived as a continuous metadata flow which would then be utilised by the other metadata sources.

If the mediadata flow is a continuous stream, then in a recorded broadcast scenario pre-compiled metadata can either be sent along the appropriate metadata flow to arrive ahead of the relevant mediadata, or held back to be transmitted in near synchronisation with the media stream. In both these cases, the framework needs to employ buffering to overcome the lag between the media and metadata, and to provide resilience against any network jitter. In the first case the receiving end of the metadata flow must buffer the data, and in the second the source buffers instead. Alternatively, in a live broadcast scenario the metadata would normally be created as the mediadata is sent, so there will always be a minimal processing delay which causes the metadata to lag behind. Here either the mediadata source must be buffered awaiting the readiness of the metadata, or the receiving end of the metadata must buffer the mediadata

instead.

In all cases, the greater the coordination of the the flow, the smaller the need for buffering becomes. A greater reliance on coordination in turn requires more timely and reliable delivery of the metadata flow. Unlike many forms of real-time multimedia where data can be dropped or scaled back to compensate for network congestion, lost metadata cannot be replaced through interpolation. So the transport mechanism for metadata flows must be real-time and **reliable** - protocols such as RTP are totally unsuitable.

4.3.2 Presentation

We will refer to the point at which a user views and uses a combination of media and metadata flows as a *presentation point*. This is a deliberate avoidance of client / server terminology since it will become apparent that within this framework a ”client” to one ”server” can be a ”server” to another - it is not necessarily the final exit from the framework of a continuous metadata flow, but the point at which one particular user views it as part of a distributed multimedia application. There is no reason why a presentation point should only be the convergence of a *single* mediadata and metadata flow; it should pull together and synchronise as many metadata flows as the user requests. Since a mediadata flow is the timer against which other flows are synchronised, any metadata flow usefully displayed at a presentation point must have been derived from that originating mediadata at some point. Multiple presentation points for multiple mediadata flows can exist on one machine, for one user, at the same time, but they should be dealt with as separate entities within the framework.

The presentation mechanism also starts to place requirements on the information the framework must encode in the metadata flow (in addition to the metadata itself):

- The metadata must have an identifying mechanism so that it can be associated with other metadata from the same flow in a consistent manner.

It should also allow the identification of the mediadata flow against which the metadata makes its assertions. It might take the form of a unique code, or could be an assertion relative to the media. These are crucial for any further processing, synchronisation, and display.

- To synchronise the media and metadata, each metadata assertion to be applied in the context of the media must have a pair of validity timestamps bounding when the metadata is true in relation to that mediadata and the timing information embedded within it. Not all metadata has a temporal dimension, but that which does must be scoped in it - since the metadata is transient, and the data it is augmenting is continually changing over time, the assertion made in the metadata needs to be placed on the temporal axis.

- For user presentation there should be another pair of timestamps bounding an extension around the valid time, during which it is is suggested that the metadata is displayed. Although the metadata makes an assertion about the mediadata which is true for a particular period, this may need to be extended by a metadata author so that it is visible at the presentation point for a longer length of time; this could obviously be overridden by user or application presentation preferences.

- To present and interpret the metadata in a suitable manner there must be a code to describe the content type of the payload the metadata packet is carrying. This might already be encoded within the metadata, e.g. an OWL ontology specified for RDF content. Although the means of identifying the content type needs to be standardised within the framework, or at least understood by the nodes that process the metadata, the format of the content itself need not be.

4.3.3 Filters

For each mediadata flow within the framework there might be several accompanying metadata flows from various sources; by selecting a particular source, or set of sources, an application can tailor the content it receices. The flexibility provided by the separation of metadata from mediadata flows can be greatly enhanced through the introduction of additional nodes in the network between the metadata source and the presentation point, nodes which process the metadata to produces derivative and supplementary flows.

These *filter* nodes are distributed throughout the framework, taking one metadata flow as their input, modifying the input metadata in some way, and outputting the modified metadata as a new metadata flow. The output of one node can be linked to the input of another so that the end result of metadata processing between source and presentation is formed from a series of simpler, more specialised, processing steps within the framework, thus extending the concept of the Microcosm filter chains [52]. Each filter is expected to perform a relatively specific form of processing, and by doing so it can be located at a point where the resources it may require are best available. As a result of this, individual metadata flows within the framework should carry specific types of metadata payloads to allow maximum flexibility between filters. A filter should not have to demultiplex a metadata flow so it can select only relevant data. Separation of the metadata from the mediadata flow means that many filter nodes will not need to receive the original mediadata flow, conserving network resources which would otherwise be needed to deliver the mediadata to every node (see figure 4.1).

Filter nodes introduce inevitable delays in the delivery of metadata from source to presentation point. In minimising this delay we clarify the question of buffering presented earlier: in the default case, to give the filters as great a margin of time as possible for processing the flow, metadata should leave the source ahead of the corresponding mediadata epoch whenever possible. For mediadata, buffering is often introduced at presentation points to reduce the

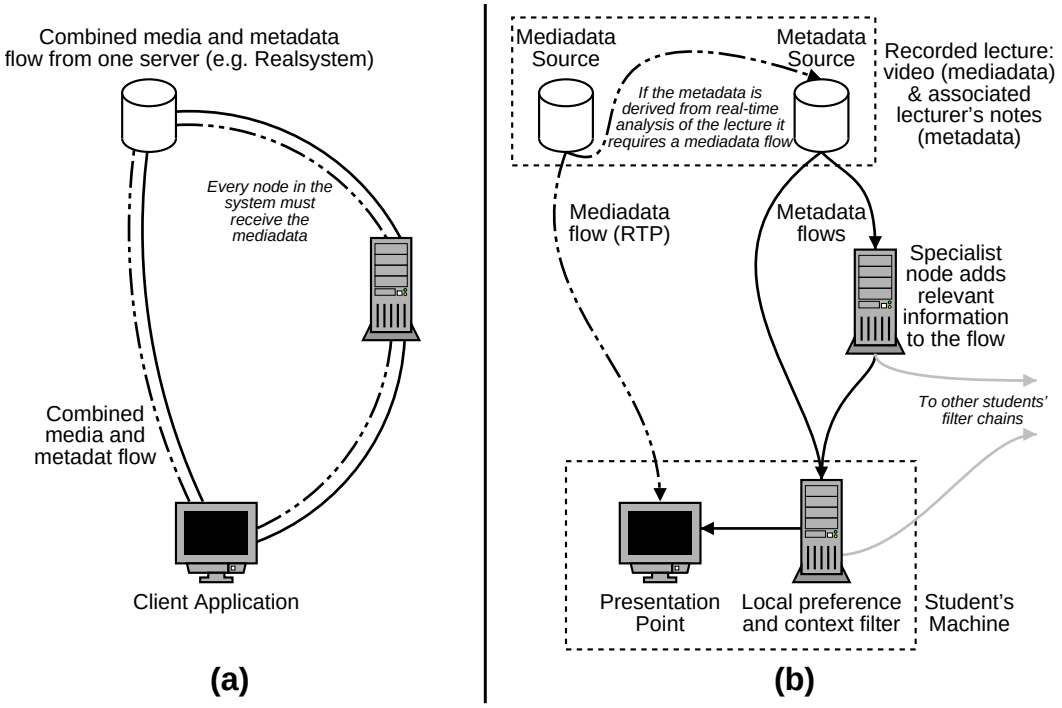

FIGURE 4.1: (a) A system in which media and metadata flows are combined; (b) A simple framework system (based on a unicast realisation of the scenario in 3.4.3)

effects of network jitter, and any leeway this provides should be absorbed as temporal capacity to the advantage of filter nodes. In some situations this might lead to metadata arriving at a presentation point before the mediadata, in which case it must be buffered too.

The overall effect of a filter should be to either add or subtract metadata from that which a user receives and makes use of at a presentation point. To add data, the filter output flow can be synchronised with the original metadata flow at the presentation point. To remove, or truly filter, the metadata, the filter output should be the only flow accepted at the presentation point: the original flow will be dropped. To accommodate this, metadata flow identities must incorporate the notion of derivatives, such that the history of a flow can be traced back through filters to the original metadata source identity. For specific applications, it may be useful to prescribe presentational relationships between metadata flows (e.g. flow x must be presented with flow y, but should not be presented with flow z), and these also need to be encoded in the metadata flow.

4.3.4 Control

Even with buffering at the presentation point, network congestion could delay metadata which needs fixed synchronisation with other flows; in this situation the stalled flow can either be dropped by the application if timeliness of presentation is paramount, or the remaining flows can be paused while waiting for a resumption of data. Further suspensions or temporal realignments of the metadata flow are likely to be triggered by application specific features and user operation - as a user navigates and pauses mediadata the metadata flows must respond accordingly (especially in a non-broadcast scenario).

To provide such functionality within the framework, mechanisms must exist to relay control messages from presentation points to the filters and sources that feed it, and between the various filters in a processing chain. There are two generic approaches to propagating the control messages:

1. Send the control message from the presentation point directly to the media and metadata sources, then allow propagation of the commands from one filter in the chain to the next, progessing back towards the presentation point. This takes advantage of any heirachical redundancy used to efficiently distribute the media and metadata, but is most effective in broadcast scenarios when a set of presentation points expect to recive an identical flow of information.

2. Send the control message from the presentation point to all the filters one hop ”upstream” of the presentation point, then each filter propagates the message up through the filter chain towards the source, one hop at a time, as far as is necessary.

The suitability of these approaches is dependent on the application in use and the topology of the inter-connected metadata flows - in particular whether they are unicast or, as discussed in the next section, multicast.

4.4 Developing the Framework for Multicast Flows

One of the requirements of distributed multimedia applications is to support group communication, and to effectively fulfil this condition the framework should also support multicast connections. Not only does this allow more efficient distribution of broadcast-style continuous metadata through lower duplication, it also increases the flexibility with which the framework can direct and combine flows from one node to others, simplifying the mechanism for richer functionality in filter chains.

4.4.1 Sources, Flows, and Presentation Points

While the media and metadata sources themselves remain largely unchanged, when the data they produce is output into the framework it is transmited using *multicast* flows. A source will send a particular media or metadata flow to any number of filter nodes or presentation points simultaneously, where all the nodes recieving the flow are members of a multicast group. In a live broadcast or group presentation environment all the nodes receiving a flow require the same (media or meta) data at the same point in time, so the use of multicast simplifies the management of the data transfer and makes more efficient use of the network. If a node wishes to start receiving a flow it can join the relevant multicast group - there is no need to allocate and set up a new point-to-point unicast flow. Since metadata must be delivered with a high degree of reliability (section 4.1), the framework now has a requirement for some form of reliable multicast (section 2.2.3).

In the case of a single presentation point requesting a flow there is no inherent advantage in using a multicast connection rather than point-to-point, although even in this scenario communication between filter nodes may be better served by multicast, since several secondary processing nodes could receive the multicast

output of a single filter earlier in the chain.

4.4.2 Filters

To receive a metadata flow from another filter or source, a filter node only has to join the multicast group that flow is being sent to. In turn the filter can easily transmit its output to many other filters or presentation points using the same mechanism. This use of multicast can create a much more sophisticated web of interrelated filter chains available to presentation points; it is much simpler for a node to join or branch a filter chain without duplicating the resources before it.

Multicasting filters also increases the scalability of the processing service offered by each node. For example, in a live video broadcast a hypermedia server may identify relevant sections of the picture using image processing techniques. It would be an inefficient use of both the hypermedia server and network resources for the many clients who receive the video stream to query the server individually; using the framework the hypermedia server only processes the video once, then the results are multicast through a metadata flow to as many presentation points as requested.

4.4.3 Control

In many ways expanding the distribution of framework flows (from one-to-one to one-to-many) is simplified through the use of multicast. To receive a flow (from a source or filter) a presentation point or filter can just join the multicast group the flow is being transmitted on; this is a much more elegant solution than maintaining state about multiple point-to-point links both for the whole framework, and for individual nodes.

Other facets of control inevitably become more complex with the introduction of group communication. A single source or filter can be expected to control the flow of media or metadata to many other nodes, and any control messages to or

from these nodes regarding stalling, restarting, and jumping to new points in a flow must be dealt with.

Since a flow is inherently temporal, with a limited period of validity, a node might request a timing re-alignment of continuous metadata which cannot be accomodated by the current instanciation of the filter chain - the flow is time shifted beyond what can be compensated for by buffering. If this offset is not required by all the nodes being fed data from a filter or source, it will need to branch the flow: the original stream will continue, but a new additional flow must be initiated in alignment with the requested epoch.

A forked flow of continuous metadata, while derivative and processed by an identical function (at any one filter node), is independently transmitted through the framework and should be considered a new and separate flow. When a subsequent filter chain is constructed branched metadata should be available for inclusion in the same way as for any other flows.

Considering again the two control dissemination mechanisms (section 4.3.4, above):

1. Control messages sent directly to the source can easily and efficiently be propagated to later nodes in the chain using multicast, in parallel with the flow of continuous metadata. This is particularly effective if the control signal has common relevance to all filters and presentation points, for example a video clip being replayed to a distributed class or meeting. However, if the message is only intended to modify the flow received by a small number nodes at the end of a chain, the mechanism must allow leading nodes to ignore the directive whilst forwarding it on.

2. Control messages sent from a presentation point or filter to the immediately preceeding node are more suitable if the result of the request is a fork in the chain by way of initiating a new flow (because of larger than buffer size difference). How a node contstructs this flow will be application and implementation dependent, which will determine whether control

message are propagated upstream.

4.5 Scenarios in the framework

It is useful to examine one of the scenarios from chapter 3 in terms of the elements within the conceptual framework.

Figure 4.2 shows part of the live news broadcasting scenario as it might be structured using multicast continuous metadata flows and framework elements. The mediadata and metadata flows are transmitted between nodes using multicast; to receive a particular flow a node joins the same multicast group the as the preceeding node in the chain (which is broadcasting the data).

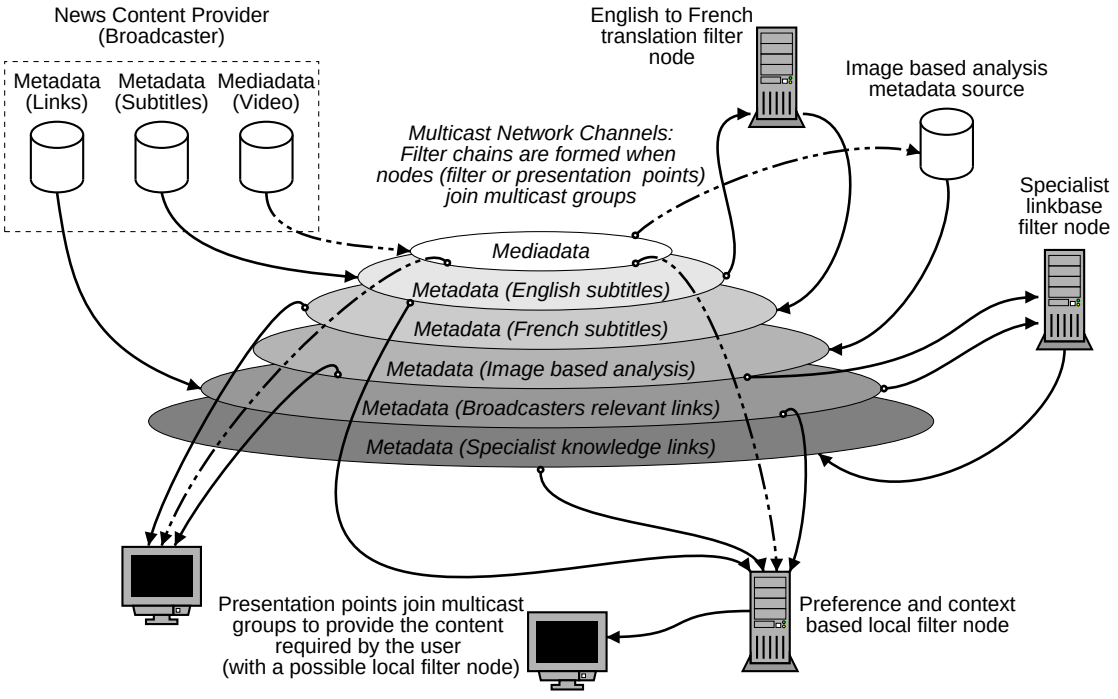

FIGURE 4.2: Live news broadcast scenario

The broadcaster would provide a mediadata stream containing the audio and video transmission of the news programme, live from the studio. The broadcaster would also be the source of two continuous metadata flows, one containing subtitles, and the other relevant links (perhaps to related archive stories on the

broadcasters web site). Each of these is carried via a separate multicast group (channel), and any other node that wishes to receive these flows (filter nodes and presentation points) must also joins these groups.

There is no requirement for all nodes to subscribe to the mediadata group - end users (each being a presentation point) would certainly wish to receive the media stream, but an automated English to French translation filter node would only need to process the subtitles metadata flow. This processed metadata flow is output onto another multicast channel, and is transferred independently to any presentations points (or filter nodes later in the chain) that subscribe to it, where it is resynchronised with the mediadata flow for presentation.

Similarly, a news aggregating service might analyse the link flow provided by the broadcaster (perhaps mediating between the original subject matter and their own information store using a web ontology) to provide a further continuous metadata feed of related links and stories from other publishers.

On the other hand, an independent filter node performing image based analysis would only subscribe to the mediadata stream, as it does not have a direct need for any of the other existing metadata flows. This node would generate a new metadata multicast ouput flow by processing the raw media directly, perhaps monitoring the video stream for shapes or images it has been trained to recognise.

When all the flows reconverge at a presentation point, users might need to cull the metadata received down to a digestible level, or speculatively subscribe to many flows then intelligently build up personalised content; this would be performed by a filter node at the same location as the presentation point.

4.6 Summary of Framework Requirements

In summary, the proposed framework has the following definitive features:

- Metadata is continuous and traverses the framework in a temporally

significant manner.

- The framework contains three types of node: sources, filters, and presentation points.
- Mediadata is the flow against which other metadata flows are synchronised. It would normally be the temporal multimedia stream the metadata is derived from.
- Metadata is carried through the framework in separate flows to the mediadata, so that intermediate (filter and presentation) nodes need only receive and process the media or metadata flows they require.
- Transmission of media and metadata between nodes should be multicast where possible.
- Metadata flows must be carried by a reliable transport.
- The metadata carried within the flow (its payload) can be in any recognised metadata format, but in addition the flow:
  - must encapsulate a mechanism for identifying the flow with regard to its relation with the mediadata flow it should be associated and synchronised with, and any derivation from a source metadata flow.
  - must encode a means for bounding the validity of temporally significant metadata in relation to the mediadata.
  - may encode a means for bounding the validity of payload metadata for the purposes of display at the presentation point.
  - must encapsulate a mechinism to identify the type of metadata in the payload so it can be decoded.
- Filter nodes perform processing on an incoming metadata flow and output the results in another (amending the identifying and derivative mechanisms appropriately). The output from one filter node can be chained to the input of another (or many other) filter nodes to create a filter chain.

- Control mechanisms between nodes should manage the rate of transmission and buffering of the temporal flows.

Chapter 5

Experience using Metadata for Distributed Multimedia

The CoAKTinG (Collaborative Advanced Knowledge Technologies in the Grid) project was conceived with the aim of aiding distributed collaboration though the integration and evolution of several existing software tools, with the addition of new tools and techniques to further enhance the collaborative experience. The use of metadata to exchange and structure information gathered by the tools was central to this integration, and builds upon the notion of continuous metadata introduced in the preceding chapters [36]. Distributed collaboration, as explored in CoAKTinG, is a key motivational scenario for this thesis (3.4.4); the project provided valuable practical experience and validation in the use of continuous metadata. This chapter presents the core elements of CoAKTinG, the use of metadata within the project, and evaluates the effectiveness of these approaches.

5.1 Introduction to CoAKTinG and the Semantic Grid

While Grid computing is often thought of in terms of providing a distributed system of high-performance compute resources, this is only one aspect required

when supporting successful use of Grid Computing. The Grid must also provide structured access to the wealth of data produced and held within it, and an environment within which the collaborative processes of investigation can occur - be this meetings between researchers, or shared access to experiments. This Grid as a composite of computational grid, data grid, and collaborative grid can be defined in terms of dynamic virtual organisations [71]:

> The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations. The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering.

Collaboration has been supported for some time on the Grid through large-scale video conferencing systems such as the Access Grid [138]. CoAKTinG set out to enhance these capabilities at the intersection of Grid collaboration and the *Semantic Grid* [73].

The Semantic Grid vision is to employ Semantic Web technologies *within* the Grid, as part of the internal machinery of the Grid (as opposed to using Semantic Web technologies atop the Grid - a knowledge grid). The use of Semantic Web technologies to integrate the foundation tools described below brings this notion of the Semantic Grid to the collaborative aspects of the Grid.

The concepts explored in earlier chapters are directly applicable to the collaborative grid and were incorporated into the CoAKTinG project. Collaboration as an activity can be seen as a resource in itself with associated metadata, and which with the right tools can be used to enhance and aid future collaboration and work.

5.2 CoAKTinG Metadata Sources

The foundations of the CoAKTinG toolset are formed by three pre-existing applications which were further developed within the project framework. Individually they have proved successful performing specific tasks supporting collaboration [111]; collectively they provide *metadata sources* to augment the *mediadata* produced in distributed collaboration - this second provision is the focus of discussion in this section, in which the metadata potential of each tool is listed.

5.2.1 BuddySpace

Overview

Developed at the Open University, BuddySpace [65, 140] is an Instant Messaging (IM) environment extended to enhance awareness of *presence* - the availability, readiness, and capability for communication of collaborating users. It is the most inherently distributed of the CoAKTinG foundation tools, being a client/server implementation of the widely used Jabber instant messaging protocol [126]. The server includes functionality for intelligent service discovery and automatic generation of a user's roster ('buddy list') derived from metadata about their involvement in particular collaborations (e.g. group membership, role, location), such that the user can quickly access relevant members of the collaboration without having to manually create contact lists. The client messaging interface is extended to present a graphical visualisation of collaboration members and their presence on an image, geographical, or conceptual map (figure 5.1).

In the distributed audio/visual meeting scenarios of CoAKTinG, BuddySpace provides a less intrusive textual 'backchannel' for communications and sharing of electronic references (e.g. URLs, documents), and meeting management (agenda queuing, electronic voting).

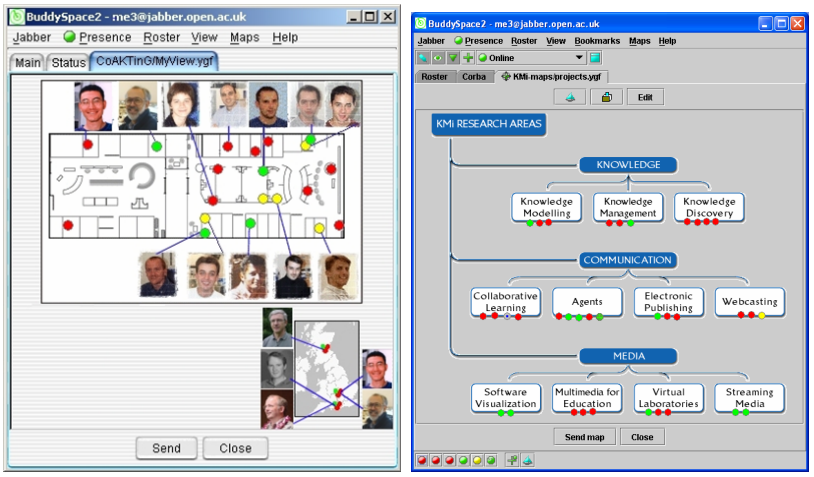

FIGURE 5.1: BuddySpace client with geographical and conceptual presence maps (from [111])

Metadata Sources

BuddySpace can provide an electronic record of the presence of collaborators in a distributed meeting, be that their location, or position in an organisational or project structure. If logging is enabled on the server, any information conveyed through the instant messaging system can be indexed on a temporal timeline. Of these, natural language communication, while more structured than audio/video data, cannot be perfectly automatically mapped to higher level information (though techniques do exist); other messages can be explicitly or implicitly typed, including presence status changes, URLs, and voting, which can provide directly accessible metadata about the meeting.

5.2.2 Compendium

Overview

Compendium, also developed at the Open University, is a tool for creating and publishing *Dialogue Maps*, which are used to capture the flow and structure of knowledge generated by a collaboration - it acts as a kind of "group memory caputure". The user transcribes Issues, Ideas, and Arguments as nodes within a graphical hypertext between which links can be asserted, as well as links to

external objects such as documents, multimedia objects, and URLs [46] (figure 5.5). Compendium also supports several more advanced hypertext features: typed links, transclusion, labelled tagging of nodes, and managed catalogues of nodes.

Maps can be based upon pre-defined Issue Templates, automatically generated by other compliant software, or captured free-form as discussion takes place. In a meeting situation the latter approach is more likely to be adopted, with one participant acting as a scribe for the group; if the evolving map is visible to all participants on a shared display it can also help focus discussion, and ensure a commonly agreed record is produced.

Metadata Sources

Compendium has a rich hypertext heritage, and while the set of node types is limited - Questions, Ideas, and Arguments - a highly structured record of discussion is created by the links between these nodes. For any scribe using Compendium while a meeting progresses, there will always be a practical limit to the amount if information that can be captured, and a trade-off between the textual description stored in a node and the hypertext surrounding it. The map is unlikely to be fully complete, and is perhaps more comparable to a set of minutes than an exhaustive and authoritative metadata record. Using Compendium in this manner does, however, usually mean that a node is created or modified while the discussion is taking place, so the editing timestamps Compendium stores for the nodes and links can be used to map Compendium structure into metadata for more detailed mediadata recordings of the meeting.

5.2.3 I-X Process Panels

Overview

Process Panels, developed at the University of Edinburgh, provide users with an interface to the I-X system [137]. At the simplest level, Process Panels can be viewed as an intelligent 'to-do' lists, and in the context of CoAKTinG are used for activity management and guidance. This might include step-by-step assistance in setting up group communication, through structuring routine or periodic administrative meetings, to tracking and co-ordinating project tasks as they are discussed and followed through in, and after, meeting discussions.

A Process Panel presents each user with an individually tailored perspective on the underlying activity or task; in a collaborative environment this activity will be shared, and each Process Panel displays to its user the steps they need to take to complete the overall task, or the issues which are blocking the progress of their actions. Activities can be decomposed, refined, delegated (to other users through their Panels), 'standard operating procedures can be incorporated, and automated agents invoked to perform sub-tasks.

Metadata Sources

The I-X system has at its heart the <I-N-C-A> ontology [136], a shared representation of synthesis tasks, in which the processes and products comprising the task are represented by abstract nodes which are related by constraints, and about which issues are generated and resolved. This approach builds upon a significant body of AI experience in planning, scheduling, process, workflow, and activity management.

While the extent of formalising an activity within the I-X framework can be set at a level appropriate for the given task, that which is formalised can be represented using the <I-N-C-A> ontology, and as such provides an excellent source of structured metadata for the activity. Although some of the activities

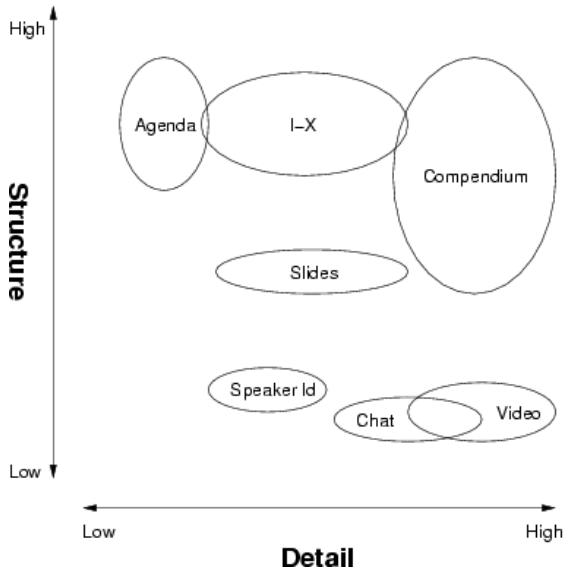

FIGURE 5.2: Relative levels of detail and structure for metadata and mediadata sources (from [111])

assisted by Process Panels may take place during a collaborative meeting, many of them will be tasks which, while initiated at, progress checked at, or returning results to a meeting, will be performed outside of the meeting scope itself. To ensure any structure from I-X can be included in the meeting corpus, Process Panels can send a summary of the activity to Compendium, where it can be included in the meeting map.

5.3 CoAKTinG Metadata and the Meeting Replay Tool

A typical meeting will involve significant amounts of information being presented by, and exchanged between, participants. Furthermore, the nature of discussion is often subtle and can be difficult to fully capture without loosing important details. An audio/video recording of the meeting will provide sufficient *detail* to capture the complete discussion, but without a navigation *structure* the quantity of data is too verbose to usefully reference. Figure 5.2 compares the levels of detail and structure provided by the media and metadata sources found in CoAKTinG.

The foundation tools described in the previous section not only assist meeting based collaboration directly, but also act as metadata sources to augment the mediadata sent between sites in a collaborative meeting environment, such as the Access Grid. In CoAKTinG, this metadata is further used to generate a hypertext which can be applied to the mediadata recording to enable indexing and navigation - this hypertext is presented through the Meeting Replay tool.

To do so the metadata is extracted from the foundation tools and mediated using a common *ontology*, and merged in a common domain - time. This is a processing, or *filter*, node in terms of the conceptual framework. As we should expect from continuous metadata, the information carried is temporally significant, and this exhibits itself though to presentation in the Meeting Replay tool, in which the navigational hypertexts primary axis is time.

The CoAKTinG Meeting Ontology

The Advanced Knowledge Technologies (AKT) project, with which CoAKTinG was affiliated, developed a reference ontology [6] to describe the domain of computer science research in the UK, as exemplified by the CS AKTive Space semantic web application [133]. Within this domain, its vocabulary is able to express relationships between entities such as individuals, projects, activities, locations, documents and publications. For purposes of capturing meeting specific information, the reference ontology is already suitable for encapsulating:

- the meeting event itself.
- meeting attendees.
- projects which are the subject matter of the meeting.
- documents associated with the meeting, including multimedia.

For activities such as meetings, which we wish to index and navigate temporally, the way in which the ontology represents time is of particular relevance. The

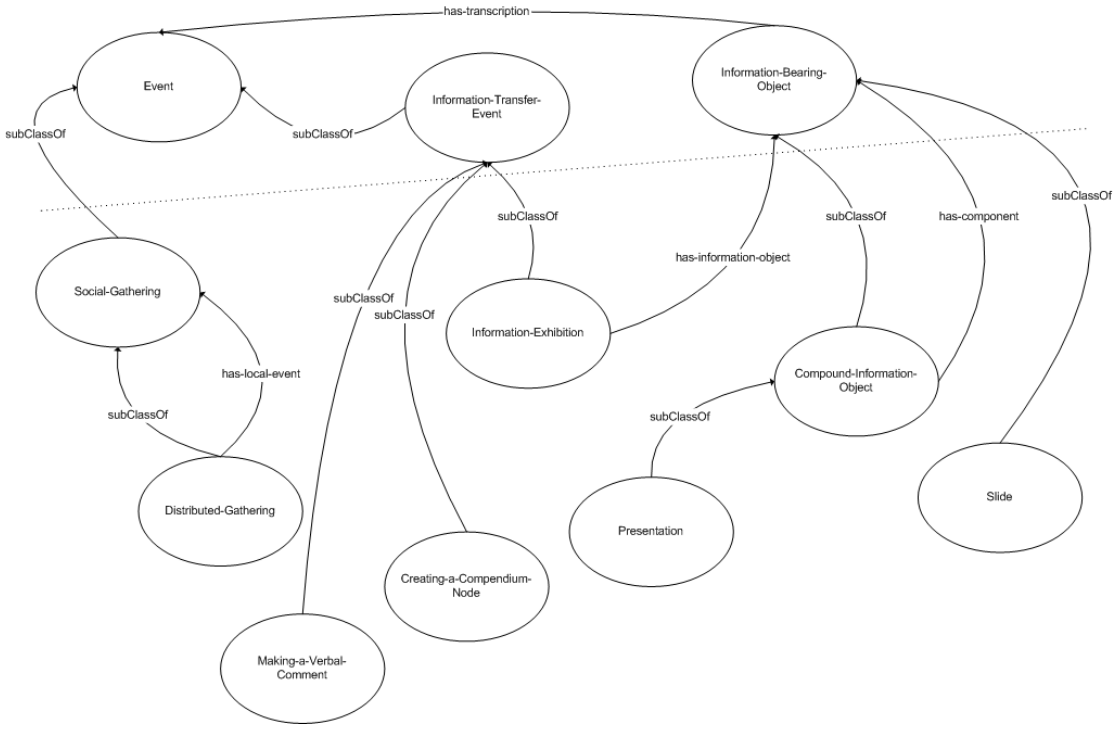

FIGURE 5.3: A simplified representation of the meeting ontology

reference ontology contains the notion of an *Event*, which is a *Temporal-Thing* that can define a duration, start and end times, a location and *agents* involved in the event. More importantly, each Event can express a *has-sub-event* relationship with any number of other Events, and it is with this property that we build up our temporal meeting structure. Within the ontology there are also many Event sub-classes, such as *Giving-a-Talk*, *Sending-an-Email*, *Book-Publishing*, and *Meeting-Taking-Place*.

While the reference ontology provides a foundation for describing meeting related resources, the CoAKTinG meeting ontology (figure 5.3, Appendix) extends the OWL version of AKT reference ontology to better encompass the concepts needed to represent collaborative spaces and activities, including:

- time properties sufficient for multimedia synchronisation.
- distributed gatherings to represent meetings which simultaneously take place in several spaces, both real and virtual.
- exhibition of information bearing objects; e.g. showing a slide as part of a

presentation.

- compound information objects; e.g. to describe a presentation consisting of several multimedia documents.
- rendering of information objects; e.g. JPEG image of a slide.
- transcription of events; e.g. a video recording of a presentation, minutes of a meeting.
- annotation of events; e.g. making a verbal comment, creating a Compendium node.

When a meeting takes place we 'mark up' the event with metadata - details such as those listed above - to build a structured description of the activities that occur. Through use of an ontology shared and understood by several different tools, we can lower the workload needed to provide usable and useful structure. Whilst there is still an overhead in gathering the metadata, it is substantially reduced by gathering it from the foundation tools already supporting collaboration.

Meeting Replay Tool

The Meeting Replay Tool is a *presentation point* - it takes metadata from the foundation tools, mediated through the ontology, and presents structure decoded from the metadata to the user, alongside the mediadata record of a meeting. The *Event/has-sub-event* structure held within the RDF is mapped onto a more conventional timeline, which is annotated with any metadata about the meeting that might aid the user in navigation. The Meeting Replay is presented using HTML and Javascript as a web site (figure 5.4) in which the user can navigate using the video timeline, or jump to a particular point by selecting one of the annotated events.

In CoAKTinG, timeline annotations included:

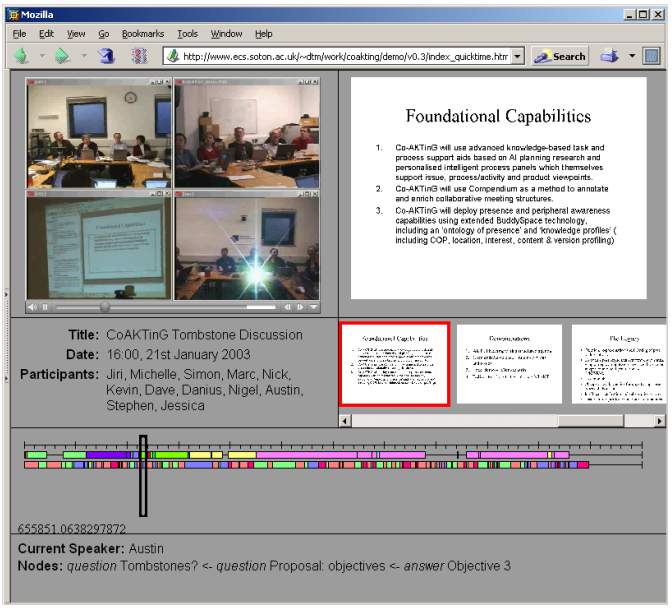

FIGURE 5.4: The Meeting Replay tool

- Agenda Items.
- Slide transitions.
- Compendium nodes.
- Speaker Identification.
- I-X activity
- BuddySpace chat

The Meeting Replay tool is also a realisation of the generalisation of link streams, and link data, to metadata through Semantic Web technologies; within the Meeting Replay hypertext are links into the video, to other multimedia objects (e.g. slide thumbnails), and other hypertext resources (e.g. web pages about people and places generated from the AKT triplestore, Compendium maps).

5.4 CoAKTinG Scenarios

Early trials of the CoAKTinG toolset focused on supporting distributed meetings of the Principal Investigators for the AKT project. The mediadata in these

experiments ranged from telephone conferences, through webcam equipped PCs, to full Access Grid sessions. These meetings were fairly well structured, with an agenda circulated some time in advance, a chair running the meeting, presentations from each site followed by brief discussion, and participants who were at least familiar with the technologies involved. Even within the limited scope of these meetings, it was seen that the metadata from the foundation tools could be used to aid the discussion as it progressed, and to create a useful hypertext record of the meeting.

5.4.1 Scientific Exploration on Mars

As part of long-term research into manned Mars missions, NASA's Work Systems Design and Evaluation group conducts annual field trials of its agent-based software and robots at the Mars Society's Desert Research Station (MDRS) in Utah, USA [43]. As a part of the 2004 trial, several CoAKTinG tools were used to support the collaboration that occurs between the astronauts on 'Mars' and the distributed groups of support scientists on Earth (known as the Remote Science Team, RST, and in this particular case specialists in geology). This was to provide CoAKTinG with a much more dynamic proving-ground, complex data flows, and the opportunity to interact with users unfamiliar with the project toolset.

The role of the RST is to analyse the data collected by the astronauts during their Extra-Vehicular Activities (EVAs) on the planet surface, and the subsequent debrief at the Mars base (which is videoed to provide a detail-rich recording). Throughout the EVA semantically annotated data is collected using the NASA agent robots. Communication delays between Earth and Mars mean that the usual means of collaboration at a distance, such as real-time conversations and the sharing of computer screens, are impractical. This is further complicated by the international composition of the RST, who collaborate across several time zones.

A typical "day" would be as follows:

1. The astronauts and their agent-based robots perform EVAs on the Martian surface, collecting samples, photos, and data for the geologists on Earth to analyse.

2. At the Mars base the astronauts debrief. Some of the days activities can be downloaded directly from the robots into Compendium maps, which form the basis for the debrief discussion. A video recording of the meeting is taken (mediadata) while Compendium is used as a dialogue mapping tool - a screen capture of Compendium use is also made (mediadata). Metadata is exported from Compendium, and any other source on the base (e.g. speaker identification).

3. The media and metadata are downloaded to Earth during a short satellite communications window. The mediadata is encoded and held on a NASA streaming server.

4. The CoAKTinG ontology is used as a mediator to produce a Meeting Replay, which is available for RST members to view before they meet as a group. RST members also receive the Compendium map of the debrief meeting to view in conjunction with the replay (figure 5.5).

5. The RST, who are distributed across several timezones, meet using telephone conferencing (mediadata). The Meeting Replay has *GroupSync* functionality, which allows all members to review a particular section of the replay together so they can discuss what they see (at the same time) over the teleconference. At any time a different RST member can take control and "direct" the replay, highlighting a particular section or issue. The RST meeting is also transcribed in a Compendium map, and the view shared throughout the meeting using screen capture technology (mediadata). Throughout the mission, and especially during their meetings, the virtual community of the RST is supported by BuddySpace.

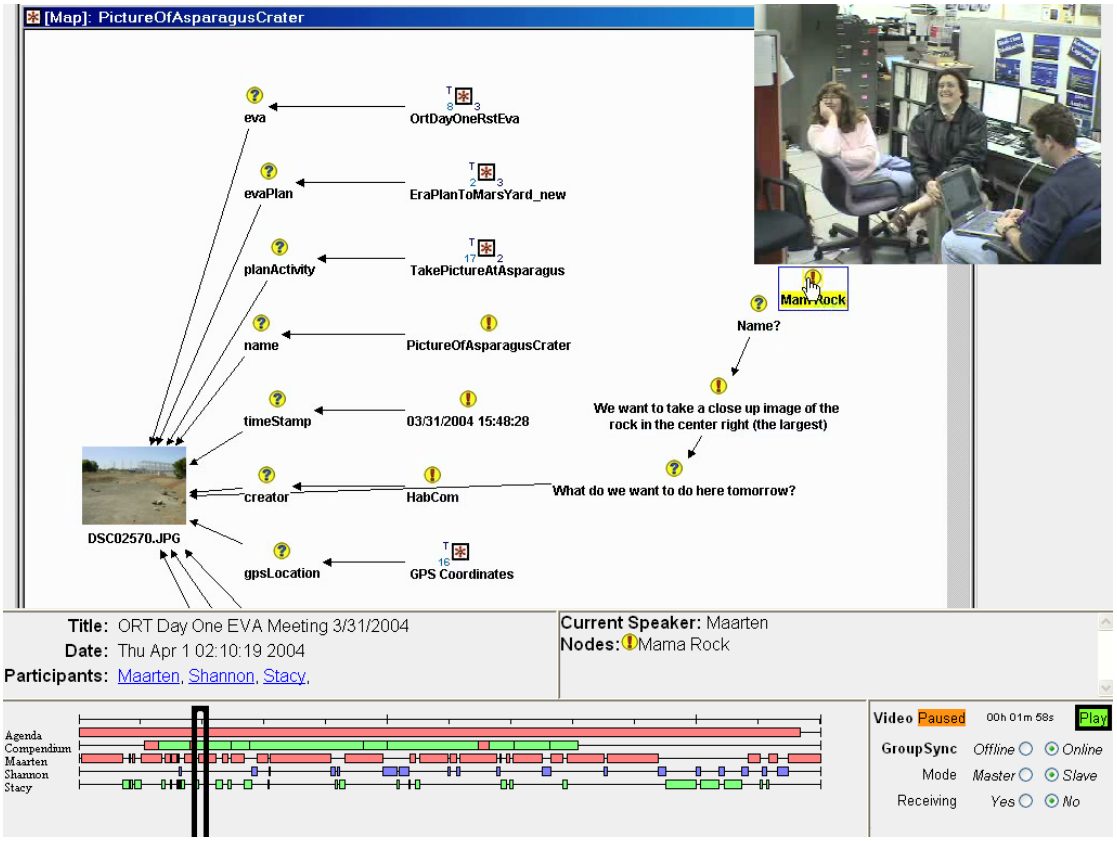

FIGURE 5.5: A Meeting Replay of the astronauts debrief as the RST would view it. The upper portion shows the mediadata, comprising a video of the astronauts and a Compendium Dialogue Map of their deliberations. The lower section includes the timeline and GroupSync control.

6. The Compendium map of the RST meeting is sent back to Mars with the RST analysis - this is used to plan for the next EVA.

GroupSync was added to the Replay Tool through extra continuous metadata flows containing synchronisation and control information sent between the Replay Tool instances using Jabber. Another metadata flow from the scribe's copy of Compendium carries the same type of information, such that debrief discussion about a particular node can be linked directly from Compendium.

5.5 Evaluation

The CoAKTinG project, and in particular the Mars Desert Research Simulation, provided a wide-ranging testbed for the ideas presented in earlier chapters. In

this section the practical realities and experiences of CoAKTinG are related to the conceptual framework for supporting continuous metadata; discussion is based around the framework requirements of section 4.6.

The experience has validated the generalisation of structure introduced in section 3.3.1. Through the mediation of the CoAKTinG ontology, diverse sources of information can be combined and filtered, in this case to create a navigational structure for the streamed mediadata record of meetings. More usefully, none of the metadata sources form, on their own, a comprehensive description of the media; by combining several sources through the mediating ontology we gain a more useful structure – the sum is greater than the parts.

5.5.1 Continuous Metadata and Temporal Significance

- *Metadata is continuous and traverses the framework in a temporally significant manner.*

Metadata in CoAKTinG is temporally significant: time is the common domain which ties together the sources of metadata from the foundation tools, and the axis along which the navigational hypertext is constructed in the Meeting Replay tool.

Metadata passed between Compendium and the multiple Meeting Replay instances during RST distributed meetings is continuous, and traverses from source to presentation points in a temporally significant manner. This element of the RST meeting is conversational. However, the down-link from "Mars" to "Earth" introduces a discontinuity in the flow of continuous metadata during the transmission stage of the lifecycle; the Meeting Replay itself could not be created continuously on-the-fly, and therefore this element of the RST meeting is presentational (and as a whole is therefore a combined distributed multimedia application). This type of discontinuity is not confined to Martian communications - the high bandwith up-link required to transfer the video data from the MDRS in the Utah desert was genuinely satellite based, and only

available for a limited period of time each day (although the time lag was considerably less than inter-planetary!).

The discontinuity raises the question as to whether the CoAKTinG RST meeting 'application' supports *live* or *stored* media, since it includes elements of both. Here we concluded, as in section 3.3, that "the mechanism [...] is of less importance than the continuousness of the metadata".

5.5.2 Framework Nodes

- *The framework contains three types of node: sources, filters, and presentation points.*

During the Mars scenario, the *media sources* were: the video of the astronauts debrief (stored then streamed); the Compendium screen capture of the astronauts debrief (stored then streamed); the Compendium screen capture during the RST meeting (live); and the teleconference channel during the RST (live). It should be noted that screen capture is a highly inefficient way to distribute the use of Compendium amongst meeting participants; in addition it can lead to lossy information transfer as the screen capture will loose resolution to cope with network congestion. Since the structure held by Compendium can be defined within the ontology, all editing within Compendium could be sent, on-the-fly, as continuous metadata, with a further flow to describe placement of nodes in the graphical map space. A distributed version of Compendium to do just this was under development, but not completed in time for the Mars trials: in such a version each copy of Compendium would be a potential metadata source *and* a presentation point.

The *metadata sources* were the foundation tools: for the Mars scenario principally Compendium, which generated metadata from both the astronaut debrief and during the RST meeting. Compendium also encapsulated domain knowledge transfered directly from the EVA robots. Speaker identification metadata was added to the astronaut debrief, although this was transcribed by

hand due to technical and budgetary constraints - this is not expected to be an issue for actual Mars missions, when each astronaut would be carrying a personal microphone as a minimum. BuddySpace was deployed to the RST, and logged for some meetings, though not used to generate Meeting Replays (RST meetings were not indexed with a Meeting Replay as originally envisaged due to time and resource constraints).

The Meeting Replay tool was the principle *presentation point* for the RST members, although through its metadata link to the Replay tool, Compendium also played a part.

The *chain of filters* was the process of mediating the metadata through the CoAKTinG meeting ontology to generate the hypertext displayed in the Meeting Replay tool. During the week long simulation, the scripts to perform these transformations were initiated by hand each day once the upload from the MDRS had been received - this experience alone should justify automation! If Compendium were distributed using continuous metadata, rather than screen capture, such automation would be more straightforward.

5.5.3 Mediadata and Separation of Structure

- *Mediadata is the flow against which other metadata flows are synchronised. It would normally be the temporal multimedia stream the metadata is derived from.*

The mediadata is the most detailed capture of the meetings: video for the astronaut debrief, and telephone for the RST. The metadata flows describe the meetings captured in the mediadata.

- *Metadata is carried through the framework in separate flows to the mediadata, so that intermediate (filter and presentation) nodes need only receive and process the media or metadata flows they require.*

The separate flows of metadata proved their utility in two specific instances:

- Due to limited bandwith on the satellite up-link from the MDRS relative to the size of the video data, and a further delay added by video encoding in the MDRS, the Compendium metadata was transfered for conversion into the Meeting Replay tool ahead of the complete video recording. This is an extreme instance of the buffering phenomena for filter nodes discussed in section 4.3.3.
- The RST members, working from many different locations including home-workers, were connected to the Internet by a variety of means. For reasons of reliability they chose to convene by telephone, a medium unsuitable for carrying continuous metadata.

5.5.4 Support for Group Communication

- *Transmission of media and metadata between nodes should be multicast where possible.*

Although multicasting was used in the early CoAKTinG Access Grid trials supporting PI meetings, neither the media nor metadata were multicast during the Mars simulation. Even though multicast would have served the scenario well - particularly in the case of synchronised viewing of the Meeting Replay during the RST meeting - the practical reality was that multicast networking was not available to the majority of the RST members. As Jabber was already used within the project for BuddySpace, the same infrastructure was used for the (unicast) transmission of metadata.

5.5.5 Metadata flows

- *Metadata flows must be carried by a reliable transport.*

Reliable transfers were used for the up-link from the MDRS, and Jabber (reliable) for real-time flows. These might not have scaled to very large groups of participants without the use of multicast.

- *The metadata carried within the flow (its payload) can be in any recognised metadata format, but must encapsulate information to synchronise and present the payload data.*

Metadata was encoded using RDF according to the CoAKTinG meeting ontology: assertions form part of an RDF graph, and other assertions within that graph have temporal scope and provide boundaries of validity for synchronisation and presentation.

Using RDF is an interesting contrast to encoding timestamps and identifiers directly within a (hypothetical) application protocol for continuous metadata (which could be comparable to the header information in RTP and encoding formats for mediadata). Although RDF has an XML representation, it should not be confused with an XML data structure or transmission protocol for continuous metadata.

As stated in section 4.2:

> Metadata [...] will be split into discrete chunks of information within the continuous metadata flow, where each is an assertion about the mediadata. While these assertions may well build upon each other to form part of a greater information structure, and assertions about the same resource may modulate over time [...]

As continuous metadata assertions flow from source nodes, filter nodes and presentation points receive and add the assertions to their graphs. Presentation points can use the temporal assertions within the graph evaluate how and when to use the metadata to augment the media stream; filters to selectively pass on, remove, or add new assertions. This is very much in the spirit of the Semantic

Web, where local triplestores are cached subsets of the available knowledge from the whole web, while the transmission of RDF re-validates continuous metadata as a generalisation of link streams [110].

Chapter 6

Multicast Transmission of Continuous Metadata

In evaluating the practical experience of using continuous metadata in CoAKTinG, and assessing the project scenarios in terms of the conceptual framework of chapter 4, three possible shortcomings are noted:

- discontinuity in the transmission of metadata
- lack of explicit timestamps at the framework layer for synchronisation and presentation
- lack of group communication support in the form of multicast

Although collaboration in the RST meeting was live, a discontinuity in the continuous metadata occurs in the transmission of the astronauts debrief from the MDRS (as discussed in section 5.5.1). The reasons for this in the Mars simulation were entirely practical: the transfer window for up-link via satellite from the MDRS was severely limited; and the version of Compendium in use did not support continuous output of structure changes.

In section 5.5.5 it was reasoned that explicit timestamps at an application protocol level were unnecessary when the continuous metadata payload

comprised of RDF containing time based assertions. While metadata exchanged during the RST was continuous and live, it would perhaps be more compelling if the structure generated during the astronaut debrief could be seen to build in a continuous manner – such that the replay, with it's timing and synchronisation constraints, could be seen to continue working under these conditions.

The lack of multicast in the simulation is, however, the most obvious. In this chapter a proof-of-concept extension to the CoAKTinG tools is developed, in which continuous metadata is transmitted using reliable multicast; RDF assertions (simulating a live flow of information from Compendium) are processed as they arrive, continually updating an RDF graph which is used to render a simplified timeline (simulating the meeting replay).

6.1 Implementation background

To aid the design process, the functionality of the various nodes within the framework that process continuous metadata (*sources*, *filters*, and *presentation points*) can be generalised in three stages as follows (figure 6.1):

1. One or more incoming continuous metadata flows deliver data to the node from other framework nodes, except in the case of *sources* (which are entry points to the framework). For each flow, the metadata must be retrieved from the network transport and loaded into a data structure the node can manipulate in a temporal manner. Source nodes encode metadata directly from production (the smart meeting room, a stored library of metadata) into the node data structure.

2. One or more incoming flows will be pooled together into a "collection" of metadata. Metadata in each collection can be processed in a number of ways:

    - analysed to produce new, derivative, metadata (normally in an application/metadata specific way).

- stored, ready for dispatch to further processing stages. This allows metadata from otherwise separate flows to be merged in a collection then processed as one.
- queued for a period of time (temporally processed).

3. Metadata held in each collection is then dispatched:

   - to the network transport, by encoding the node's internal data structure into that of the transport.
   - to a further processing stage, maintaining the node's internal data format. As such, a more complex processing chain can be built up.
   - to a user interface or other metadata consuming process, in the case of a *presentation point*.

This breakdown shows that although the different framework nodes have separate purposes, the building blocks needed for their implementation are functionally similar.

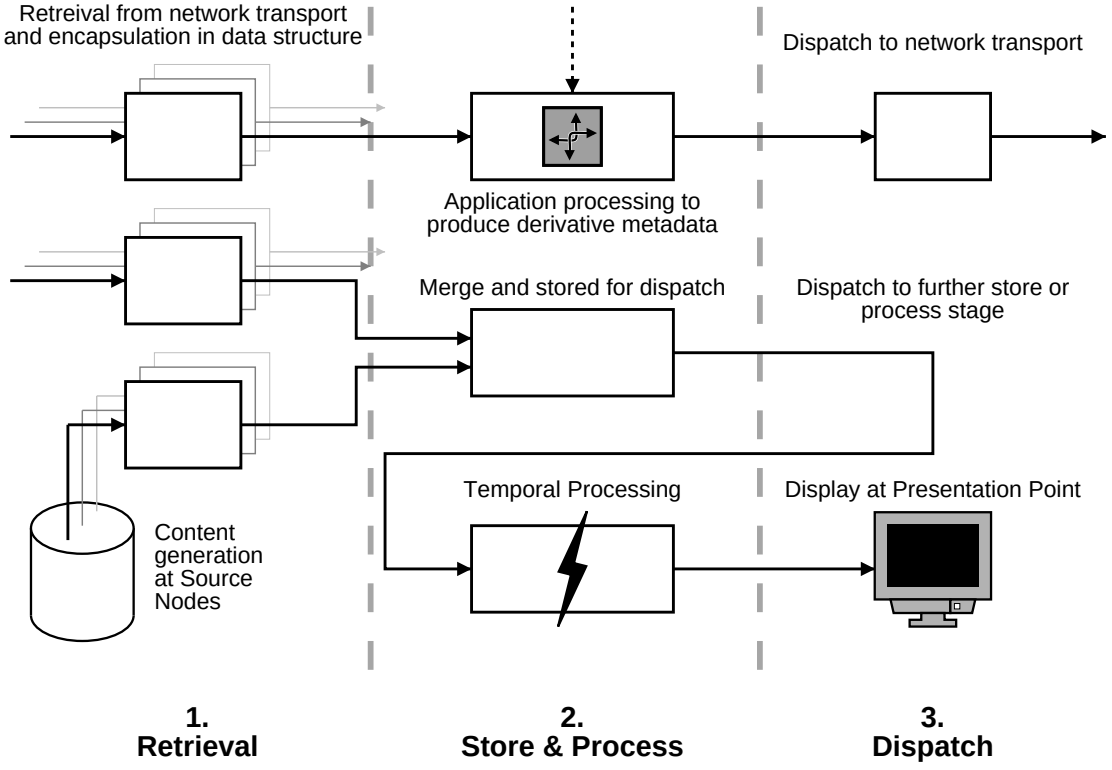

FIGURE 6.1: Three stage generalisation of framework node functionality

6.2 Early Implementation Experience

Initial framework experimentation took the form of a direct implementation of the framework requirements for metadata coupled with media streams transmitted using the Java Media Framework (JMF). During transmission, the metadata was sent as short XML documents, defined by a schema which encapsulated the framework requirements for identifying flow codes, content types, and validity timestamps for synchronisation and presentation. Within the nodes, the metadata was mapped into a series of Java classes based around the MetaEvent class (figure 6.2).

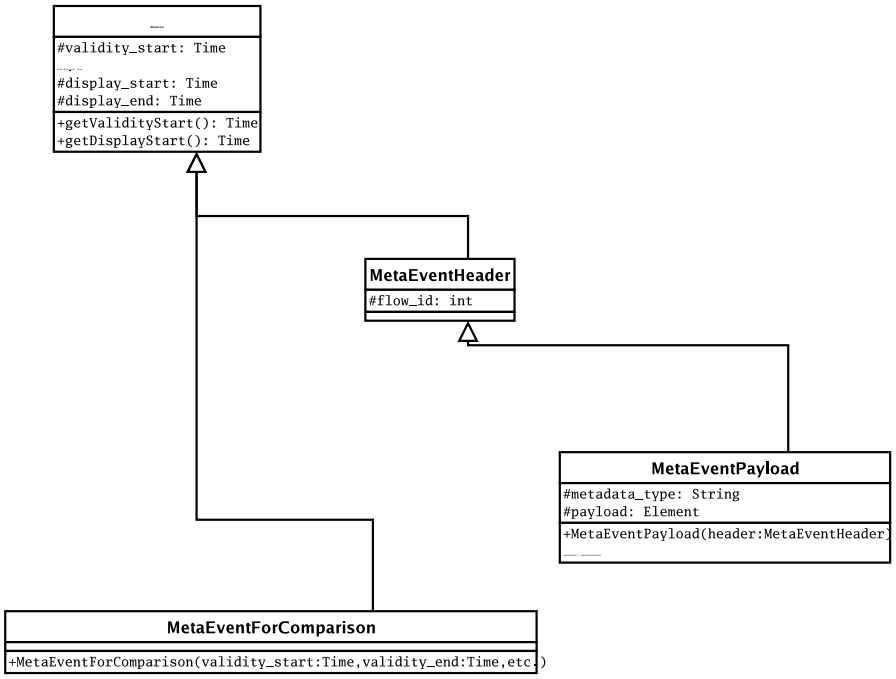

FIGURE 6.2: The MetaEvent (abstract) class and subclasses

6.3 Multicasting RDF as Continuous Metadata

While early implementations encapsulated the framework requirements directly in data structures and protocols, experience from CoAKTinG has shown that the requirements can be met with more generic node implementations, using metadata encoded with RDF where time-based assertions are common to a

shared ontology (section 5.5.5). Indeed the Semantic Web model of adding, removing, and merging sections of RDF graphs in a piecemeal way fits very well to continuous metadata, where discrete sets of assertions are forever flowing in and out of nodes. It also maps well into the generalisation for implementation introduced in section 6.1: the collections of metadata are the RDF graphs held in each node, and the nodes internal data structure is simply RDF validated by any ontologies used.

This approach of using RDF to encode metadata was therefore adopted for the proof-of-concept evaluation of multicasting continuous metadata. JGroups [14] was used to provide a reliable multicast transport, and the Jena Semantic Web Framework for Java [97] to hold an in-memory graph storing the RDF triples:

1. RDF/XML encoded assertions are sent, using a JGroups channel, to the test node using multicast. The assertions declare that a Compendium node has been created or modified - this simulates the data Compendium would send.

2. The test node receives the RDF and adds it to the graph it holds (called a model in Jena).

3. When the model changes Jena raises an event; on this event the model is queried for all temporal assertions which are then ordered into a timeline, and the subject description of the ordered time periods is rendered on a visualisation of the timeline; this represents Compendium activity.

4. The process repeats: as more assertions reach the node they are added to the graph, which is then used to re-render the timeline.

Multicast functionality is seen by instantiating several copies of the test node - all receive copies of the assertions, and all their timelines update accordingly.

While this proof-of-concept lacks the sophistication of the Meeting Replay tool and the functionality of Compendium, it verifies that the full potential of the framework could be realised if multicast networking were available to the RST,

and Compendium were enhanced to output a flow of continuous metadata directly.

Further development of the proof-of-concept would likely raise additional issues:

- assertion of time in the AKT Reference ontology (and by extension the CoAKTinG Meeting ontology) is unwieldy and verbose: each time point is separated out into seven temporal components each of which is a DatatypeProperty (year, month, day, hour, minute etc.). This is especially true when mapping events onto a temporal axis (e.g. creating the timeline) as the software must repeatedly walk through the graph. Since nodes will store large numbers of triples from multiple metadata sources, it could be expected that the graphs will become large and this inefficiency could have a serious performance implications.

- The timeline in the CoAKTinG Meeting Replay tool assumes the accompanying media is of fixed length, and thus the timeline can be too. This assumption has been also been adopted by the proof-of-concept, despite the supply of a potentially never-ending flow of continuous metadata; realistically the user interface would have to be adapted to allow a flexible amount of "scrolling zoom" showing a limited length before and ahead of the current time. Without this elements on the timeline might be shrunk to insignificant proportions.

- As detailed in section 4.3.3, without buffering or media delays inserted at presentation, metadata might not be presented until after the associated section of media.

Chapter 7

Conclusions and Future Directions

7.1 Summary of Work

The aim of this thesis has been to explore the use of metadata to enhance multimedia applications – not just at the point of authoring and storage, nor merely at presentation, but also at all points in between. This has been motivated by the need to support distributed collaboration activities, and in particular add novel real-time interactive structure to aid the user alongside streaming audio and video.

Chapter 2 surveyed the body of research surrounding the thesis, documenting how hypermedia systems have developed to incorporate temporal media, and the requirements at the network and application layers of multimedia systems. Metadata was also studied in its details and application, and from this a theme emerged: one of a generalisation of structure, represented by and in metadata, from the links of hypertext through to the ontologies of the Semantic Web.

In chapter 3 the theme of metadata as structure was examined in the context of distributed multimedia. A lifecycle perspective on multimedia distribution was introduced and used to analyse and contrast the levels of support for multimedia

and the metadata we wish to augment it with. Deficiencies in the provision of metadata through the transmission stage were identified, and continuous metadata was proposed as a means to overcome them (section 3.3); several motivational scenarios were given as illustration of how continuous metadata might be implemented and utilised.

After further analysis and justification, chapter 4 introduced a conceptual framework to define continuous metadata, and with it a set of requirements through which continuous metadata can be provided.

Continuous metadata, and the framework, were put to the test in chapter 5. The CoAKTinG project had, amongst its aims, the use of continuous metadata to enhance distributed virtual meetings, and provided an opportunity to trial the principles of this thesis in a simulation of remote group collaboration though the NASA Mars Desert Research Station.

A novel approach was taken in implementing continuous metadata by way of an OWL ontology, which was used to mediate structured temporal metadata from several different sources, serialised in RDF. This approach was validated, with continuous metadata and the framework, when a hypertext navigation was constructed from the RDF and used to enhance multimedia materials during the distributed collaboration of NASA team members.

Full multicasting of continuous metadata was not trialled during the NASA simulation, when the users did not have access to a multicast enabled network. Despite this, it was proposed that multicast would have improved the provision of metadata had it been available; this was validated by a proof-of-concept tool in chapter 6.

7.2 Future Directions

Although the CoAKTinG simulation validates the principles of this thesis, practical necessities left some aspects deserving of further enquiry, and posed

several new questions arising from experience. These are the basis for directions of future work:

Wide scale multicast trials - Although chapter 6 demonstrated a proof-of-concept validation of using multicast to distribute continuous metadata, more thorough and widespread experiments should be conducted, including the use and evaluation of different reliable multicast approaches, algorithms, and protocols.

Enhancement of tools to support real-time metadata output - the primary metadata source in the NASA simulation, Compendium, does not support real-time output of its structural metadata using RDF. There is no theoretical reason why this change could not be made; and was not completed for CoAKTinG due to the need to balance many different development priorities. As discussed in chapter 5, this would offer several benefits, not just in terms of continuous metadata, but for adding distributed functionality to Compendium itself.

Evaluation of time representation - in the CoAKTinG trials the ontology was chosen for purposes of information reuse in the context of the Semantic Web. As discussed in chapter 5, its representation of time is somewhat unwieldy. With the comparatively small data set used in the NASA simulation, it was not entirely clear how much of a performance bottleneck this may form if, for example, scaled to a triplestore with millions of assertions. Even with a more succinct representation of time, queries bound by time cannot be made elegantly through current triplestore implementations, and this issue needs further research. Could a triplestore implement specific optimisations for time assertions internally?
A performance evaluation could also be made in comparison to an implementation of the temporal requirements of the framework at an application protocol level - while this would segregate the metadata held in the domain ontology from its temporal dimensions, making manipulation and processing over the knowledge as a whole harder, would any

performance improvements make this worthwhile?

Presentation of continuous metadata - as briefly discussed in chapter 6, presentation of a live and continually evolving metadata flow has ramifications for user interface design, which should be explored. The predominant view of temporal media is of a discrete "block" within a hypermedia system - it is the synchronised presentation of the media with other hypermedia elements that has received the most attention. As such we tend to use mechanisms to jump to or from particular time indexes within the media - the timeline is superimposed so that we can synchronise hypermedia elements, as it was in CoAKTinG. The situation is further complicated when the temporal media becomes unbounded, such as in a live scenario. Here it becomes more difficult to deal with the media as a block within a carefully scripted hypermedia presentation, because the block has an potentially infinite length.

Semantic Web (Services) technologies to configure filter chains - the chain of processing for the CoAKTinG scenario - taking the metadata from source applications and, via the ontology, creating the replay interface - was configured by hand, both in terms of each step of the processing, and the overall workflow. Metadata about each element of processing, combined with the metadata in the continuous flow, could help automate a match between the desired result and the functionality available.

Evaluation of further scenarios - While CoAKTinG provided an excellent opportunity for evaluation, the other scenarios described in section 3.4 deserve further investigation. Indeed, the work in this thesis has fed into projects for e-Learning Page et al. [111] (the ELeGI project) and e-Science (Combechem).

Appendix: CoAKTinG Meeting Ontology

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE owl [
<!ENTITY owl "http://www.w3.org/2002/07/owl#">
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
<!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema#">
<!ENTITY dc "http://purl.org/dc/elements/1.1/">
<!ENTITY dct "http://purl.org/dc/terms/">
<!ENTITY support "http://www.aktors.org/ontology/support#">
<!ENTITY portal "http://www.aktors.org/ontology/portal#">
<!ENTITY ibis "http://purl.org/ibis#">
<!ENTITY meeting "http://www.aktors.org/coakting/ontology/meeting-20040304-1#">
<!ENTITY base "http://www.aktors.org/coakting/ontology/meeting-20040304-1">
]>
<!-- CoAKTinG meeting ontology added above, and as namespace below -->
<rdf:RDF xmlns:owl="&owl;"
xmlns:rdf="&rdf;"
xmlns:rdfs="&rdfs;"
xmlns:dc="&dc;"
xmlns:dct="&dct;"
xmlns:xsd="&xsd;"
xmlns:support="&support;"
xmlns:portal="&portal;"
xmlns:ibis="&ibis;"
xmlns:meeting="&meeting;"
xml:base="&base;">
```

```
<!-- Ontology definition -->
<owl:Ontology rdf:about="&base;">
<rdfs:label xml:lang="en">CoAKTinG Meeting Ontology.</rdfs:label>
<dc:title xml:lang="en">CoAKTinG Meeting Ontology.</dc:title>
<dc:description xml:lang="en">The CoAKTinG Meeting Ontology has been designed
to support the CoAKTinG project and tools, extending the AKT Reference
Ontology.</dc:description>
<dc:creator>CoAKTinG Project</dc:creator>
<dc:creator>Kevin R. Page</dc:creator>
<dct:created>2004-03-04</dct:created>
<owl:versionInfo>1.2</owl:versionInfo>
<owl:imports rdf:resource="http://www.aktors.org/ontology/portal"/>
</owl:Ontology>
<!-- ################################################################ -->
<!-- add milliseconds to TimePoints (from the AKT ontology) -->
<owl:DatatypeProperty rdf:ID="millisecond-of">
<rdfs:label xml:lang="en">millisecond of</rdfs:label>
<rdfs:domain rdf:resource="&support;Time-Point"/>
<rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:DatatypeProperty>
<!-- Extend the Time-Point definition with about -->
<owl:Class rdf:about="&support;Time-Point">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#millisecond-of"/>
<owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<!-- ################################################################ -->
<!-- a new subsclass of Event to represent meetings which
concurrently take place in several locations -->
<owl:Class rdf:ID="Distributed-Gathering">
<rdfs:label>Distributed Gathering</rdfs:label>
<rdfs:comment>Gatherings that take place in more than one physical
location.</rdfs:comment>
```

```
<rdfs:subClassOf rdf:resource="&portal;Social-Gathering"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
<!-- a Distributed Gathering must have one or more constituent
Events -->
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#has-local-event"/>
<owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="has-local-event">
<rdfs:subPropertyOf rdf:resource="&portal;has-sub-event"/>
<rdfs:domain rdf:resource="#Distributed-Gathering"/>
<rdfs:range rdf:resource="&portal;Social-Gathering"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- ################################################################# -->
<!-- Information-Exhibition is a subclass of Information-Transfer-Event, which
is used to express the exhibition / display of an Information-Bearing-Object,
e.g.  the presentation of slides or documents in a meeting -->
<owl:Class rdf:ID="Information-Exhibition">
<rdfs:label>Information Exhibition</rdfs:label>
<rdfs:comment>Information Exhibition expresses the display of an
Information-Bearing-Object, e.g.  the presentation of slides or documents in a
meeting.</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Transfer-Event"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="has-information-object">
<rdfs:domain rdf:resource="#Information-Exhibition"/>
<rdfs:range rdf:resource="&portal;Information-Bearing-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- A future version of this ontology might include an
Information-Transfer-Medium to describe the tool(s) used to
make the exhibition -->
```

```
<!-- ############################################################ -->
<!-- Define a class to describe Compound Information Objects, e.g.  a
presentation that includes multiple slides, video etc.  -->
<owl:Class rdf:ID="Compound-Information-Object">
<rdfs:label>Compound Information Object</rdfs:label>
<rdfs:comment>Compound Information Objects describe Information Bearing Objects
that are constructed from a collection of further Information Bearing Objects.
e.g.  a presentation containing several slides and a video.</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Bearing-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
<!-- We could put a cardinality constraint of "at least one" on hasComponent,
but we don't always want to list the components.  e.g.  we may not have details
or the need to express individual slides in a presentation -->
</owl:Class>
<owl:ObjectProperty rdf:ID="has-component">
<rdfs:domain rdf:resource="#Compound-Information-Object"/>
<rdfs:range rdf:resource="&portal;Information-Bearing-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- There is an argument for set ordering of the compenents of a
Compound-Information-Object (e.g.  ordering of slides in a presentation).
Obviously the order of an actual presentation is dealt with using Exhibition
(which copes with a slide being shown twice etc.), but what if a presentaion
hasn't been exhibited yet?-->
<owl:Class rdf:ID="Presentation">
<rdfs:label>Presentation</rdfs:label>
<rdfs:comment>e.g.  a PowerPoint presentation</rdfs:comment>
<rdfs:subClassOf rdf:resource="#Compound-Information-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:Class>
<owl:Class rdf:ID="Slide">
<rdfs:label>Slide</rdfs:label>
<rdfs:comment>A slide within a presentation</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Bearing-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:Class>
<!-- ############################################################ -->
<!-- has-rendered-uri allows us to describe multiple renderings of an I-B-O,
```

```
e.g.  the JPEG exports of individual slides, perhaps at different resolutions.
Further description of the rendering is currently not included in the ontology
(and could be handled by the web server with MIME types etc.)  -->
<owl:ObjectProperty rdf:ID="has-rendered-uri">
<rdfs:label xml:lang="en">has rendering</rdfs:label>
<rdfs:comment xml:lang="en">The rendering of an Information Bearing Object.
e.g.  a JPEG rendering of a Slide.</rdfs:comment>
<rdfs:domain rdf:resource="&portal;Information-Bearing-Object"/>
<!-- rdfs:range rdf:resource="&portal;Information-Bearing-Object"/ -->
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- ################################################################ -->
<!-- We need to be able to take an Event (e.g.  a Social-Gathering / meeting
and assert that an I-B-O is a record (e.g.  minutes, Compendium map, video
recording) of that Event -->
<owl:ObjectProperty rdf:ID="has-transcription">
<rdfs:label xml:lang="en">has transcription</rdfs:label>
<rdfs:comment xml:lang="en">The transcription of an event, e.g.  the minutes of
a meeting, or the video recording of a presentation.</rdfs:comment>
<rdfs:domain rdf:resource="&portal;Event"/>
<rdfs:range rdf:resource="&portal;Information-Bearing-Object"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- ################################################################ -->
<!-- The following are "temporal annotations" on the meeting.  There is a fine
line in differentiating between an annotation made during the meeting on the
meeting, and an event which is an intrinsic part of the meeting itself.  e.g.
the case is clearer for a verbal comment being an annotation, compared to
someone accessing an online resource using a web browser, or sending an instant
message.  As such, there is no temporal annotation superclass; each is an
individual subclass of Information-Transfer-Event -->
<!-- ################################################################ -->
<!-- Verbal comment Event -->
<owl:Class rdf:ID="Making-a-Verbal-Comment">
<rdfs:label>Verbal Comment</rdfs:label>
<rdfs:comment>An Event to bind when a Person makes a comment (e.g.  in a
meeting).</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Transfer-Event"/>
```

```
<rdfs:isDefinedBy rdf:resource="&base;"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&portal;sender-of-information" />
<owl:allValuesFrom rdf:resource="&portal;Person" />
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<!-- ################################################################ -->
<!-- The IBIS ontology does not have a single overarching class that will
correspond to all Compendium nodes.  So we'll create one here which comprises
ibis:Idea, ibis:Note and ibis:Reference -->
<owl:Class rdf:ID="IBIS-Concept">
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:resource="&ibis;Idea" />
<owl:Class rdf:resource="&ibis;Reference" />
<owl:Class rdf:resource="&ibis;Note" />
</owl:unionOf>
</owl:Class>
<!-- ################################################################ -->
<!-- The action of creating a compendium node (in a meeting) is separate from
the actual nodes representing meeting structure.  Thus, we are recording the
information we know about - when a Compendium node was authored - rather than
information about that node itself - which we should expect to be handled by
the IBIS ontology.  We then infer that because a node was created at a
particular time, this was the subject of discussion at that point in the
meeting.  -->
<owl:Class rdf:ID="Capturing-an-IBIS-Concept">
<rdfs:label xml:lang="en">Recording the moment represented by an IBIS
idea</rdfs:label>
<rdfs:comment xml:lang="en">This event marks when an IBIS idea is transcribed
e.g.  when compendium is used to minute a meeting.</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Transfer-Event"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&portal;sender-of-information" />
<owl:allValuesFrom rdf:resource="&portal;Person" />
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&meeting;has-ibis-node" />
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
<owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&meeting;has-ibis-map" />
<owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="has-ibis-node">
<rdfs:label xml:lang="en">has IBIS node</rdfs:label>
<rdfs:comment xml:lang="en">A Compendium node being created.  Currently the
resource is expected to be within the XML output from Compendium, rather than a
class/instance in the knowledge base.</rdfs:comment>
<rdfs:domain rdf:resource="&meeting;Capturing-an-IBIS-Concept"/>
<rdfs:range rdf:resource="#IBIS-Concept"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has-ibis-map">
<rdfs:label xml:lang="en">has IBIS map</rdfs:label>
<rdfs:comment xml:lang="en">The Compendium view the node is being created or
modified within.</rdfs:comment>
<rdfs:domain rdf:resource="&meeting;Capturing-an-IBIS-Concept"/>
<rdfs:range rdf:resource="&ibis;Map"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- ################################################################ -->
<!-- As for capturing, but these are modification events
-->
```

```
<owl:Class rdf:ID="Modifying-an-IBIS-Concept">
<rdfs:label xml:lang="en">Recording the moment when an IBIS concept is
modified</rdfs:label>
<rdfs:comment xml:lang="en">This event marks when an IBIS idea is
modified.</rdfs:comment>
<rdfs:subClassOf rdf:resource="&portal;Information-Transfer-Event"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&portal;sender-of-information" />
<owl:allValuesFrom rdf:resource="&portal;Person" />
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&meeting;has-ibis-node" />
<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
<owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&meeting;has-ibis-map" />
<owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="has-ibis-node">
<rdfs:label xml:lang="en">has IBIS node</rdfs:label>
<rdfs:comment xml:lang="en">A Compendium node being modified.</rdfs:comment>
<rdfs:domain rdf:resource="&meeting;Modifying-an-IBIS-Concept"/>
<rdfs:range rdf:resource="#IBIS-Concept"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has-ibis-map">
<rdfs:label xml:lang="en">has IBIS map</rdfs:label>
```

```
<rdfs:comment xml:lang="en">The Compendium view the node is being created or
modified within.</rdfs:comment>
<rdfs:domain rdf:resource="&meeting;Modifying-an-IBIS-Concept"/>
<rdfs:range rdf:resource="&ibis;Map"/>
<rdfs:isDefinedBy rdf:resource="&base;"/>
</owl:ObjectProperty>
<!-- ///////////////////////////////////////////////////////////////////// -->
</rdf:RDF>
```

Bibliography

[1] *Proceedings, The 11th ACM Conference on Hypertext and Hypermedia*. ACM, 2000. ISBN 1-58113-227-1.

[2] *Proceedings, The Ninth International World Wide Web Conference*. IW3C2, 2000.

[3] S. Acharaya and B. Smith. An experiment to characterize videos on the world wide web. In *Proceedings of Multimedia Computing and Networking*, pages 166–179. SPIE / ACM, 1998.

[4] R. L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

[5] Phil Adcock, Nigel Davies, and Gordon Blair. Supporting continuous media in open distributed systems architectures. *Lecture Notes in Computer Science*, 731:179–191, 1993.

[6] AKT Reference Ontology. http://www.aktors.org/ontology/, 2004.

[7] K. C. Almeroth. The evolution of multicast: From the mbone to inter-domain multicast to internet2 deployment. *IEEE Network, Special Issue on Multicasting*, 14(1):10–20, 2000.

[8] Kenneth M. Anderson. Integrating open hypermedia systems with the world wide web. In [22], pages 157–166.

[9] Kenneth M. Anderson. Data scalability in open hypermedia systems. In [139], pages 27–36.

[10] Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead. Chimera: Hypertext for hetrogeneous software environments. In *Proceedings, 1994 ACM European Conference on Hypermedia Technology*. ACM, 1994.

[11] Helen Ashman. Electronic document addressing - dealing with change. *ACM Computing Surveys*, 32(3):201–202, 2000.

[12] Cristina Aurrecoechea, Andrew T. Campbell, and Linda Hauw. A survey of QoS architectures. *ACM Multimedia Systems Journal*, 6(3):138–151, 1998.

[13] Michelle S. Bachler, Simon J. Buckingham Shum, David C. De Roure, Danius T. Michaelides, and Kevin R. Page. Ontological mediation of meeting structure: Argumentation, annotation, and navigation. In *First International Workshop on Hypertext and the Semantic Web*. 2003.

[14] Bela Ban. Design and implementation of a reliable group communication toolkit for java. 1998.

[15] Liam J. Bannon and Kjeld Schmidt. CSCW: Four characters in search of a context. In *Proceedings of the First European Conference on CSCW*, pages 3–16. 1989.

[16] A. Basso, G. L. Cash, and M. R. Civanlar. Transmission of MPEG-2 streams over non-guaranteed quality of service networks. In *Picture Coding Symposium (PCS-97)*. IEEE, 1997.

[17] Richard Beales, Don Cruickshand, David De Roure, Nick Gibbins, Ben Juby, Danius T. Michaelides, and Kevin Page. The pipeline of enrichment: Supporting link creation for continuous media. In *Lecture Notes in Computer Science / Hypermedia: Openness, Structural Awareness, and Adaptivity (International Workshops OHS-7, SC-3 and AH-3)*, volume 2266, pages 47–58. Springer-Verlag, 2001.

[18] Sean Bechhofer, Mike Dean, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Pater F. Patel-Schneider, Guus

Schreiber, and Lynn Andrea Stein. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/, 2003. Viewed 25/02/2006.

[19] Tim Berneres-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, 1992.

[20] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): Generic syntax. http://www.ietf.org/rfc/rfc2396.txt, 1998. Viewed 25/02/2006.

[21] Tim Berners-Lee. *Weaving the Web: The Past, Present and Future of the World Wide Web*. Texere Publishing, 1999.

[22] Mark Bernstein, Leslie Carr, and Kasper Østerbye, editors. *Proceedings, The 8th ACM Conference on Hypertext*. ACM SIGLINK, 1997. ISBN 0-89791-866-5.

[23] Steven G. Blackburn and David C. De Roure. A tool for content based navigation of music. In [64], pages 361–368.

[24] J. H. Blair. Supporting cooperative work with computers: addressing meeting mania. In *Proceedings of the 34th IEEE Computer Society International Conference*, pages 208–217. IEEE, 1989.

[25] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. http://www.ietf.org/rfc/rfc2475.txt, 1998. Viewed 25/02/2006.

[26] J-C. Bolot. End-to-end delay and loss behavior in the internet. In *Proceedings of ACM SIGCOMM*, pages 289–298. ACM, 1993.

[27] Luca Bompani and Fabio Vitali. Providing hypertextual functionalities with XML. In [1], pages 214–215.

[28] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading style sheets, level 2: Css2 specification. http://www.w3.org/TR/CSS2/, 1998. Viewed 25/02/2006.

[29] Niels Olof Bouvin. Unifying strategies for web augmentation. In [139], pages 91–101.

[30] Niels Olof Bouvin and René Schade. Integrating temporal media and open hypermedia on the World Wide Web. In [99], pages 375–378.

[31] John Bowler, Milt Capsimalis, Richard Cohn, David Dodds, Andrew Donoho, David Duce, Jerry Evans, Jon Ferraiolo, Scott Furman, Brent Getlin, Peter Graffagnino, Rick Graham, Vincent Hardy, Lofton Henderson, Alan Hester, Bob Hopgood, Dean Jackson, Christophe Jolif, Kelvin Lawrence, Chris Lilley, Philip Mansfield, Kevin McCluskey, Tuan Nguyen, Troy Sandal, Peter Santangeli, Haroon Sheikh, Gavriel State, Robert Stevahn, Timothy Thompson, and Shenxue Zhou. Scalable vector graphics (SVG) 1.0 specification. http://www.w3.org/TR/2000/CR-SVG-20000802/, 2000. Viewed 25/02/2006.

[32] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. http://www.ietf.org/rfc/rfc1633.txt, 1994. Viewed 25/02/2006.

[33] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) - version 1 functional specification. http://www.ietf.org/rfc/rfc2205.txt, 1997. Viewed 25/02/2006.

[34] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998. Viewed 25/02/2006.

[35] M. Celia Buchanan and Polle T. Zellweger. Specifying temporal behavior in hypermedia documents. In *Proceedings, ACM European Conference on Hypertext ECHT'92*, pages 262–271. ACM SIGLINK/SIGWEB, 1992.

[36] 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 at the Eleventh IEEE Int. Symposium on High Performance Distributed Computing. IEEE, 2002.

[37] Stephan Bugaj, Dick Bulterman, Bruce Butterfield, Wo Chang, Guy Fouquet, Christian Gran, Mark Hakkinen, Lynda Hardman, Peter Hoddie, Klaus Hofrichter, Philipp Hoschka, Jack Jansen, George Kerscher, Rob Lanphier, Nabil Layaïda, Stephanie Leif, Sjoerd Mullender, Didier Pillet, Anup Rao, Lloyd Rutledge, Patrick Soquet, Warner ten Kate, Jacco van Ossenbruggen, Michael Vernick, and Jin Yu. Synchronized multimedia integration language (SMIL) 1.0 specification. http://www.w3.org/TR/REC-smil/, 1998. Viewed 25/02/2006.

[38] Dick Bulterman, Guido Grassel, Jack Jansen, Antti Koivisto, Nabil Layaïda, Thierry Michel, Sjoerd Mullender, and Daniel Zucker. Synchronized multimedia integration language (SMIL 2.1). http://www.w3.org/TR/2005/REC-SMIL2-20051213/, 2005. Viewed 25/02/2006.

[39] Vannevar Bush. As we may think. *The Atlantic Monthly*, pages 101–108, 1945.

[40] Les A. Carr, David W. Barron, Hugh C. Davis, and Wendy Hall. Why use HyTime? *Electronic Publishing: Origination, Dissemination and Design*, 7(3):163–178, 1994.

[41] Les A. Carr, David C. DeRoure, Wendy Hall, and Gary J. Hill. The Distributed Link Service: A tool for publishers, authors and readers. In *Fourth International World Wide Web Conference: The Web Revolution*, pages 647–656. IW3C2, 1995.

[42] Deming Chen, Regis Colwell, Herschel Gelman, Panos K. Chrysanthis, and Daniel Mossé. A framework for experimenting with QoS for multimedia services. In Martin Freeman, Paul Jardetzky, and M. Harrick, editors, *Multimedia Computing and Networking 1996*, pages 186–197. SPIE, 1996. ISBN 0-8194-2041-7.

[43] W.J. Clancey, M. Sierhuis, R. Alena, D. Berrios, J. Dowding, J.S. Graham, K.S. Tyree, R.L. Hirsh, W.B. Garry, A. Semple, S.J. Buckingham Shum, N. Shadbolt, and S. Rupert. Automating CapCom Using Mobile Agents and Robotic Assistants. In *Proceedings of 1st AIAA Space Exploration Conference*. 2005.

[44] James Clark and Steve DeRose. XML path language (XPath) version 1.0. http://www.w3.org/TR/xpath, 1999. Viewed 25/02/2006.

[45] J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, 1987.

[46] J. Conklin, A. Selvin, S. Buckingham Shum, and M. Sierhuis. Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS. In *Proceedings The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01)*, pages 123–124. 2001.

[47] P. Cotter and B. Smyth. Ptv: Intelligent personalised tv guides. In *Proceedings of the 12th Innovative Applications of Artificial Intelligence (IAAI-2000)*, pages 957–965. AAAI, 2000.

[48] Ismail Dalgic and Hanlin Fang. Comparison of H.323 and SIP for IP telephony signaling. In *Proceedings of Photonics East*. SPIE, 1999.

[49] Ron Daniel Jr., Steve DeRose, and Eve Maler. XML pointer language (XPointer) version 1.0. http://www.w3.org/TR/xptr, 2000. Viewed 25/02/2006.

[50] Duco Das and Herman Horst. Recommender systems for tv. In *Proceedings of the AAAI Workshop on Recommender Systems*, pages 35–36. AAAI, 1998.

[51] Hugh C. Davis. To embed or not to embed. *Communications of the ACM*, 38(6):108–109, 1995.

[52] Hugh C. Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Robert J. Wilkins. Towards and integrated information environment with open

hypermedia systems. In *Proceedings, ACM European Conference on Hypertext ECHT'92*, pages 181–190. ACM SIGLINK/SIGWEB, 1992.

[53] David C. De Roure, Don G. Cruickshand, Danius T. Michaelides, Kevin R. Page, and Mark J. Weal. On hyperstructure and musical structure. In *The Thirteenth ACM Conference on Hypertext and Hypermedia (Hypertext 2002)*. ACM, 2002.

[54] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. http://www.ietf.org/rfc/rfc2460.txt, 1998. Viewed 25/02/2006.

[55] Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, 1990.

[56] Steven E. Deering. *Multicast Routing in a Datagram Network*. Ph.D. thesis, Stanford University, 1991.

[57] Edmund X. DeJesus. Toss your tv: How the internet will replace broadcasting. *Byte Magazine*, 1996.

[58] Steve DeRose, Eve Maler, David Orchard, and Ben Trafford. XML linking language (XLink) version 1.0. http://www.w3.org/TR/xlink/, 2000. Viewed 25/02/2006.

[59] G. DeSanctis and B. Gallupe. A foundation for the study of group decision support systems. *Management Science*, 33(5):589–609, 1987.

[60] P. Deutsh. GZIP file format specification version 4.3. http://www.ietf.org/rfc/rfc1952.txt, 1996. Viewed 25/02/2006.

[61] S. Draper, H. Earnshaw, E. Montie, S. Parnall, R. Toll, D. Wilson, and G. Winter. TV Anytime. In *Proceedings International Broadcasting Convention (IBC 99)*, pages 103–108. IBC, 1999.

[62] DVB Project. Digital Video Broadcasting resource page. http://www.dvb.org/, ???? Viewed 25/02/2006.

[63] Wolfgang Effelsberg and Thomas Meyer-Boudnik. MHEG explained. *IEEE Multimedia*, 2(1):26–38, 1995.

[64] Wolfgang Effelsberg and Brian C. Smith, editors. *Proceedings, The 6th ACM Conference on Multimedia*. ACM SIGMULTIMEDIA, 1998.

[65] M. Eisenstadt, J. Komzak, and M. Dzbor. Instant messaging + maps = powerful tools for distance learning. In *Proceesings of TelEduc03*. 2003.

[66] W. C. Elm and D. D. Woods. Getting lost: A case study in user interface design. In *Proceedings of the Human Factors Society 29th Annual Meeting*, pages 927–931. Human Factors Society, 1985.

[67] D. C. Engelbart. A conceptual framework for the augmentation of man's intellect. *Vistas of Information Handling*, 1, 1963.

[68] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol - HTTP/1.1. http://www.ietf.org/rfc/rfc2068.txt, 1997. Viewed 25/02/2006.

[69] Roy T. Fielding, E. James Whitehead Jr., Kenneth M. Anderson, Gregory A. Bolcer, Peyman Oreizy, and Richard N. Taylor. Web based developmetn of complex information products. *Communications of the ACM*, 41(8):84–92, 1998.

[70] S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *ACM SIGCOMM Computer Communication Review*, 25(4):342–356, 1995. ISSN 0146-4833.

[71] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[72] G. Ghinea and J.P. Thomas. QoS impact on user perception and understanding of multimedia video clips. In [64], pages 49–54.

[73] C.A. Goble, D. De Roure, N. Shadbolt, and A.A.A. Fernandes. Enhancing services and applications with knowledge and semantics. In C. Kesselman and I. Foster, editors, *The Grid 2: Blueprint for a New Computing Infrastructure (2nd edition)*. Morgan-Kaufmann, 2004.

[74] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. HTTP extensions for distributed authoring - WEBDAV. http://www.ietf.org/rfc/rfc2518.txt, 1999. Viewed 25/02/2006.

[75] S. Goose and W. Hall. The development of a sound viewer for an open hypermedia system. *The New Review of Hypermedia and Multimedia*, 1:213–231, 1995.

[76] Kaj Grønbæk, Niels Olof Bouvin, and Lennert Sloth. Designing dexter-based hypermedia services for the world wide web. In [22], pages 146–156.

[77] Kaj Grønbæk, Jens A. Hem, Ole L. Madsen, and Lennert Sloth. Designing dexter-based cooperative hypermedia systems. In *Proceedings, Fifth ACM Conference on Hypertext (Hypertext '93)*, pages 25–38. ACM, 1993.

[78] Kaj Grønbæk, Lennert Sloth, and Niels Olof Bouvin. Open hypermedia as user controlled meta data for the web. In [2], pages 554–566.

[79] J. Grudin. CSCW: Its history and participation. *IEEE Computer*, 27(5):19–26, 1994.

[80] Abdelhakim Hafid, Gregor von Bochmann, and Rachida Dssouli. Distributed multimedia applications and quality of service: A review. *Electronic Journal on Network and Distributed Processing*, 1998.

[81] Frank Halasz and Mayer Schwartz. The dexter hypertext reference model. In Judi Moline, Daniel Benigni, and J. Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 95–133. National Institute of Standards and Technology, 1990.

[82] Brent Halsey and Kenneth M. Anderson. XLink and open hypermedia systems: A preliminary investigation. In [1], pages 212–213.

[83] M. Handley, J. Crowcroft, C. Bormann, and J. Ott. Very large conferences on the Internet: The Internet Multimedia Conferencing Architecture. *Computer Networks*, 31(3):191–204, 1999.

[84] Lynda Hardman. *Modelling and Authoring Hypermedia Documents*. Ph.D. thesis, Amsterdam University, 1998.

[85] Lynda Hardman, Dick C. A. Bulterman, and Guido van Rossum. The amsterdam hypermedia model: Adding time and context to the dexter model. *Communications of the ACM*, 37(2):50–62, 1994.

[86] Lynda Hardman, Jacco van Ossenbruggen, K. Sjoerd Mullender, Llyod Rutledge, and Dick C.A. Bulterman. Do you have the time? Composition and linking in time-based hypermedia. In [139], pages 189–196.

[87] Gary J. Hill and Wendy Hall. Extending the microcosm model to a distributed environment. In *Proceedings, 1994 ACM European Conference on Hypermedia Technology*, pages 32–40. ACM, 1994.

[88] Jane Hunter. Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*. 2001.

[89] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML encryption syntax and processing. http://www.w3.org/TR/xmlenc-core/, 2002. Viewed 25/02/2006.

[90] ITU-T. Recommendation H.323, packet-based multimedia communications systems. 1998.

[91] ISO/IEC JTC1/WG4. Smdl draft international standard (iso/iec dis 10743). 1995.

[92] Gail Kaiser and Jim Whitehead. Distributed authoring and versioning. *IEEE Internet Computing*, 2(5):34–40, 1998.

[93] Frank Kappe, Gerald Pani, and F. Schnabel. The architecture of a massively distributed hypermedia system. *Internet Research*, 3(1):10–24, 1993.

[94] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax/, 1999. Viewed 25/02/2006.

[95] John J. Legget and John L. Schnase. Viewing dexter with open eyes. *Communications of the ACM*, 37(2):77–86, 1994.

[96] H. Lieke and D. Suciu. XMill: An effiecient compressor for xml data. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. ACM, 2000.

[97] B. McBride. Jena: Implementing the RDF Model and Syntax Specification. In *Proceedings of the WWW2001 Semantic Web Workshop*. 2001.

[98] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proceedings, the 3rd ACM Conference on Multimedia*, pages 511–522. ACM, 1995.

[99] A. Mendelzon, editor. *Proceedings, The Eighth International World Wide Web Conference*. IW3C2, 1999.

[100] Dave Millard, Luc Moreau, Hugh Davis, and Sigi Reich. Fohm: A fundamental open hypertext model for investigating interoperability between hypertext i domains. In [1], pages 93–102.

[101] David E. Millard. Discussions at the data border: From generalised hypertext to structural computing. *Journal of Network and Computer Applications, Special Issue on Structural Computing*, 25(4):296–310, 2002.

[102] Eric Miller. An introduction to the Resource Description Framework. *D-Lib Magazine*, 1998.

[103] Frank Nack and Adam T. Lindsay. Everything you wanted to know about MPEG-7: Part 1. *IEEE Multimedia*, 6(3):65–77, 1999.

[104] Frank Nack and Adam T. Lindsay. Everything you wanted to know about MPEG-7: Part 2. *IEEE Multimedia*, 6(4):64–73, 1999.

[105] Theodore H. Nelson. Literary machines. Published by the author, 1981.

[106] Jakob Nielsen. The art of navigatin through hypertext. *Communications of the ACM*, 33(3):296–310, 1990.

[107] Peter J. Nürnberg and Helen Ashman. What was the question? Reconciling open hypermedia and the world wide web research. In [139], pages 83–90.

[108] Peter J. Nürnberg, John J. Legget, and Uffe K. Wiil. An agenda for open hypermedia research. In Kaj Grønbæk, Elli Mylonas, and Frank M. Shipman, editors, *Hypertext '98*, pages 198–206. ACM SIGLINK, 1998.

[109] Peter J. Nürnberg, John J. Leggett, and Erich R. Schneider. As we should have thought. In [22], pages 96–101.

[110] Kevin R. Page, Don Cruickshank, and David De Roure. Its about time: Link streams as continuous metadata. In *The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01)*, pages 93–102. ACM, 2001.

[111] Kevin R. Page, Danius T. Michaelides, Simon J. Buckingham Shum, Yun-Heh Chen-Burger, Jeff Dalton, David C. De Roure, Marc Eisenstadt, Stephen Potter, Nigel R. Shadbolt, Austin Tate, Michelle Bachler, and Jiri Kozmak. Collaboration in the Semantic Grid: a Basis for e-Learning. *Applied Artificial Intelligence*, 19(9-10):881–904, 2005.

[112] Pengkai Pan and Glorianna Davenport. I-Views: A community-oriented system for sharing streaming video on the internet. In [2], pages 567–582.

[113] S. Paul, K.K. Sabnani, J.C.H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, 1997. ISSN 0733-8716.

[114] Amy Pearl. Sun's link service: A protocol for open linking. In *Proceedings, The 2nd ACM Conference on Hypertext (Hypertext '89)*, pages 137–146. ACM, 1989.

[115] Steven Pemberton, Murray Altheim, Daniel Austin, Frank Boumphrey, John Burger, Andrew W. Donoho, Sam Dooley, Klaus Hofrichter, Philipp Hoschka, Masayasu Ishikawa, Warner ten Kate, Peter King, Paula Klante, Shin'ichi Matsui, Shane McCarron, Ann Navarro, Zach Niesk, Dave Raggett, Patrick Schmitz, Sebastian Schnitzenbaumer, Peter Star, Chris Wilson, Ted Wugofski, and Dan Zigmond. XHTML 1.0: The extensible hypertext markup language. http://www.w3.org/TR/xhtml1/, 2000. Viewed 25/02/2006.

[116] Maria Pimental, Gregory Abowd, and Yoshide Ishiguro. Linking by interacting: A paradigm for authoring hypertext and hypermedia. In [1], pages 39–48.

[117] J. Postel. User datagram protocol. http://www.ietf.org/rfc/rfc0768.txt, 1980. Viewed 25/02/2006.

[118] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 specification. http://www.w3.org/TR/html4/, 1999. Viewed 25/02/2006.

[119] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of IEEE INFOCOM*, pages 680–688. IEEE, 1994.

[120] Injong Rhee. Error control techniques for interactive low-bit rate video transmission over the internet. In *Proceedings of ACM SIGCOMM*, pages 290–301. ACM, 1998.

[121] Tom Rodden. A survey of cscw systems. *Interacting with Computers*, 3(3):319–354, 1991.

[122] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol. http://www.ietf.org/rfc/rfc3261.txt, 2002. Viewed 25/02/2006.

[123] Lloyd Rutledge, Lynda Hardman, and Jacco van Ossenbruggen. Evaluating SMIL: Three user case studies. In Dick C. A. Bultermann, Kecin Jeffray,

and Hong-Jiang Zhang, editors, *Multimedia '99*, pages 171–174. ACM SIGMULTIMEDIA, 1999.

[124] Lloyd Rutledge, Lynda Hardman, and Jacco van Ossenbruggen. The use of SMIL: Multimedia research currently applied on a global scale. In *Proceedings of Multimedia Modeling 99 (MMM 99)*, pages 1–17. Computer Graphics Society, 1999.

[125] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick C.A. Bulterman. Anticipating SMIL 2.0: The developing cooperative infrastructure for multimedia on the web. In [99], pages 343–352.

[126] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. http://www.ietf.org/rfc/rfc3820.txt, 2004. Viewed 25/02/2006.

[127] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. http://www.ietf.org/rfc/rfc1889.txt, 1996. Viewed 25/02/2006.

[128] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). http://www.ietf.org/rfc/rfc2326.txt, 1998. Viewed 25/02/2006.

[129] Henning Schulzrinne. Internet services: From electronic mail to real-time multimedia. In Klaus Franke, Uwe Hübner, and Winfried Kalfa, editors, *Proceedings of KIVS (Kommunikation in Verteilten Systemen)*, pages 22–34. 1995.

[130] Henning Schulzrinne. When can we unplug the phone and the radio? In Thomas D. C. Little and Riccardo Gusella, editors, *Proceedings of the International Workshop on Network and Operating System Support for for Digital Audio and Video (NOSSDAV)*, pages 183–184. 1995.

[131] Henning Schulzrinne and Jonathan Rosenberg. Internet telephony: Architecture and protocols - an IETF perspective. *Computer Networks and ISDN Systems*, 31(3):237–255, 1999.

[132] H. A. Schütt and N. A. Streitz. Hyperbase: A hypermedia engine based on a relational database management system. In A. Rizk, N. A. Streitz, and J. Andre, editors, *Proceedings of the European Conference on Hypertext ECHT'90*, pages 95–108. 1990.

[133] N. R. Shadbolt, N. Gibbins, H. Glaser, S. Harris, and m. c. schraefel. CS AKTive Space, or How We Learned to Stop Worrying and Love the Semantic Web. *IEEE Intelligent Systems*, 19(2):41–47, 2004.

[134] Jason W. Smith, David Stotts, and Sang-Uok Kum. An orthogonal taxonomy for hyperlink anchor generation in video streams using ovaltine. In [1], pages 11–18.

[135] S. Smoliar. A computer aid for schenkerian analysis. *Computer Music Journal*, 4(2):41–59, 1980.

[136] A. Tate. <I-N-C-A>: an Ontology for Mixed-initiative Synthesis Tasks. In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems (MIIS) at the International Joint Conference on Artificial Intelligence (IJCAI-03)*. Acapulco, Mexico, 2003.

[137] A. Tate, J. Dalton, and J J. Stader. I-P^2 - Intelligent Process Panels to Support Coalition Operations. In *Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002)*. Toulouse, France, 2002.

[138] The Access Grid Project. http://www.accessgrid.org/, ???? Viewed 25/02/2006.

[139] Klaus Tochtermann, Jörg Westbomke, Uffe K. Wiil, and John J. Leggett, editors. *Proceedings, The 10th ACM Conference on Hypertext and Hypermedia*. ACM SIGWEB, 1999. ISBN 1-58113-064-3.

[140] Y. Vogiazou, M. Eisenstadt, M. Dzbor, and J. Komzak. From buddyspace to cititag: Large-scale symbolic presence for community building and

spontaneous play. In *Proceedings of the ACM Symposium on Applied Computing*. 2005.

[141] W3C. The World Wide Web Consortium resource page. http://www.w3.org/, ???? Viewed 25/02/2006.

[142] G. K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[143] G. Wei, L. Agnihotri, and N. Dimitrova. Tv program classification based on face and text processing. In *IEEE Multimedia and Expo 2000*, pages 1345–1348. IEEE, 2000.

[144] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery. http://www.ietf.org/rfc/rfc2413.txt, 1998. Viewed 25/02/2006.

[145] J. Wroclawski. The use of RSVP with IETF integrated services. http://www.ietf.org/rfc/rfc2210.txt, 1997. Viewed 25/02/2006.