Its About Time: Link Streams as Continuous Metadata

Kevin R. Page, Don Cruickshank and David De Roure
Intelligence, Agents, Multimedia, Department of Electronics & Computer Science
University of Southampton, SO17 1BJ, UK
{krp,dgc,dder}@ecs.soton.ac.uk

ABSTRACT
As enabling technologies become available there is an increasing use of temporal media streams, such as audio and video, within a hypertext context. In this paper we present the rationale and requirements for delivering continuous metadata alongside the media stream, and focus on linking as our case study. We consider the mechanism for delivery of the metadata across a distributed system, the format and content of the metadata flow itself, and the presentation of the media and augmenting metadata to the user. Two initial proof of concept applications have been developed to demonstrate these concepts, which we describe. Finally we propose a framework for highly distributed delivery and processing of multicast continuous metadata, as a part of the infrastructure necessary to provide a more complete multimedia environment for hypermedia systems.

KEYWORDS: metadata, streamed media, open hypermedia, temporal linking

INTRODUCTION

As hypermedia systems have grown, they have developed into rich multimedia environments incorporating many formats other than text and still images into the user's experience. Increasingly prevalent (and attractive to the user) amongst these formats are streamed media such as audio and video. Such streaming applications normally take one of three forms: pre-stored media (presentational media-on-demand); live broadcast media; and live interactive media (video-conferencing). The transmission mechanism can be one-to-one (unicast) or one-to-many (multicast).

How well can current solutions deliver such content? Adcock *et al.* [Adc93] identify four resources required to support distributed multimedia applications:

1. Explicit support for streamed (or temporal) media
2. The ability to specify and reserve a required Quality of Service (QoS)
3. Synchronisation within and between multimedia elements
4. Presentation and communication to and between *groups* of collaborating users

While no complete system or underlying network (mainly Internet) infrastructure yet fully meets these criteria, the pieces are beginning to fall into place: RTP [Sch96] and RTCP [Sch98] enable transport and control of streamed media; IPv6 [Dee98] brings promise of support for QoS architectures (such as IntServ [Bra94] and DiffServ [Bla98b]); multicast routing will be expanded to encompass the entire Internet (again, enabled by IPv6), facilitated by application-level routing; the Amsterdam Hypermedia Model [Har94] brings the notion of temporal presentation to hypertext systems; many of the ideas within it have become the basis for current and future versions of SMIL [Bug98]; and the many issues involved in synchronising communications have long been an important area of research within the multimedia community [Geo96].

In this paper we describe a further missing link—the processing and delivery of metadata accompanying the streamed media. This metadata is intended to convey all types of information relevant to the associated media, and in the next section we present the case for distributing the metadata separately from the media and on a continuous basis. We then describe two 'proof of concept' systems in which the metadata takes the form of links: firstly our experimental 'Temporal Linking Service', and secondly a development of this built within an agent environment and implementing FOHM [Mil00]. We generalise this work to present a distributed framework for continuous metadata, which is then extended to encompass multicast. Finally we illustrate the application of continuous metadata through a number of scenarios.

CONTINUOUS METADATA

Metadata facilitates the discovery, use and re-use of the vast resources of information available via the Internet. Its significance has been acknowledged in recent years through activities such as the semantic Web, which is establishing an infrastructure providing interoperability between applications exchanging machine-understandable information. Digital media is no exception, with a clear need for metadata describing both multimedia documents and broadcasts.

We can view a hyperstructure as metadata associated with a set of documents, a relationship explored in [Grø00]. In the multimedia context, we should expect to associate the hyperstructure with multimedia documents including streamed temporal media [Bou99]. It is this latter case which is our focus here, with the hyperstructure itself having a temporal dimension; for example, a link might be valid only for a given time interval in an audio stream.

By considering where we might use it, we can categorise the metadata on two axes with regard to the associated temporal media data: when it is transmitted, and how it is transmitted.

Stored multimedia data, such as an audio or video recording, is a persistent entity which can be described by its 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 [Pan00]) 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, but instead becomes 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 [Pim00]. It could also result from real-time processing of the stream, such as some form of classification, segmentation or annotation. It is sometimes acceptable to introduce a delay in such 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: a 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 the metadata cannot be guaranteed to be prepared 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 party 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. By way of example, the anchor generation system in OvalTine [Smi00] 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 as well as the people.

The evolution of multimedia technologies and standards also promotes the capture of metadata 'upstream' in the production process; e.g. shots, script, storyboard. The MPEG standards are evolving (through MPEG-7 [Nac99]) to accommodate this, and associated metadata, within a combined data stream. While there are advantages to transmitting and storing multimedia data with the metadata embedded in this way, we believe there are also situations where metadata should be handled separately and delivered synchronously.

A flow of metadata which is distinct from the multimedia data flow not only follows the open hypermedia convention of separation of links [Dav95] (where links are metadata for a document), but also allows for a much more flexible framework of distributed delivery, processing and presentation; metadata can originate and be used in different places to the multimedia content, indeed they may reach the user through different mediums. 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.

In any of the above scenarios the multimedia data flow may be unicast or multicast, and this also applies to the live metadata; however, it might not be the case that every party needs to receive identical metadata flows, and metadata processing nodes may not require the multimedia stream at all, thus a separate metadata flow is preferable.

Although the metadata we refer to is streamed, it may be better to think of it as *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. Continuous metadata need not be high volume, and there may be significant lulls between bursts of data (although the transporting connection is kept open); but the transmission timing of the metadata does have significance, and it will normally be augmenting continuous, streamed, media - henceforth known as 'mediadata'.

In this paper it is not the type 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. The classification and exchange of metadata can already be described by standards such as RDF and MPEG-7; there is no reason why the metadata 'payload' carried by continuous flows could not be encoded using these standards.

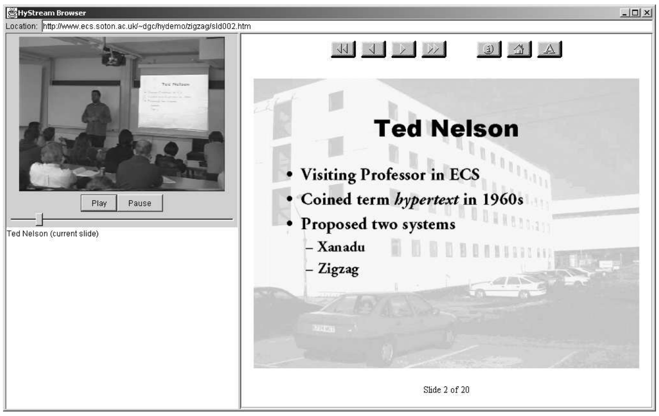

Figure 1: The TLS Client. Links from the seminar video resolve to the presentation slides in this simple scenario

THE TEMPORAL LINKING SERVICE

Our first experience of temporal linking (without streams) was the Microcosm SoundViewer [Goo95], which was subsequently extended to use RTSP in order to work with streamed media. This was used in conjunction with a number of other tools that were described in [DR98]. These tools, which were designed for media-on-demand scenarios, used time intervals in the temporal media stream as anchors; intervals could also be used to identify fragments of content from which features could be extracted for 'content-based navigation'.

Building on this experience and to demonstrate the concepts outlined in the previous section, the Temporal Linking Service (TLS) was developed to deliver continuous linking information (metadata) to hypermedia clients. Client and server applications have been developed in Java using the Java Media Framework (JMF) to stream the mediadata using RTP (a screenshot of the client is shown in figure 1). While this enables provision of applet based clients, it also limits the system to streaming formats supported by JMF. TLS metadata takes the form of links relevant to the media which the client can resolve through its associated web browser. The server retrieves the metadata from its 'linkbase' (i.e. link database) and uses XML markup to deliver it to the client. We have designed our own HTTP-like protocol to explore synchronisation issues between TLS servers and clients.

On-time delivery of metadata from the client to the server is preferred over delayed, but guaranteed, delivery. Our protocol must enable each component in the system to determine the local transmission deadline of each item of temporal metadata; we anticipate situations in which late metadata should be dropped before entering the metadata stream to increase the chances of other metadata reaching the client within time.

Protocol

The Temporal Linking Service allows a connected client to select a continuous metadata flow via some kind of descriptor (i.e. URI), and to receive a 'never-ending' linkbase relevant to that descriptor. In this paper, we do not describe how the server arranges or discerns the linkbase, but the actual protocol between a TLS aware browser and the TLS server. The Temporal Linking Transfer Protocol (TLTP) is derived from HTTP/1.1, and sends commands via a TCP socket connection; the metadata flow is maintained for the duration of the connection. A summary of commands is shown in table 1; a more extensive description of the protocol operation can be found in [Cru01].

The metadata payload (i.e. link data) is augmented with timing constraints, so that the client browser can display the links with temporal relevance. It was decided to use XML to markup this data, which led to a number of possibilities for delivering this data within the TLTP flow. For longer and more complicated quantities of metadata (and payload) it would seem sensible to use a succession of separate XML documents, which would also maintain document form and integrity should the flow be broken midway through. To maintain simplicity, however, we use multiple elements within a single XML document (a technique also used by SXML [Rog00]).

Delivery of a particular link is delayed up until a certain point which is specified by the client via the BIAS command. Temporal navigation (i.e. fast forward and rewind) within the media causes the client to issue successive BIAS commands to maintain coherency between the browser and the TLS server.

Table 2 shows an example session transcript of a client receiving metadata about a seminar presentation.

In this implementation, we note that our TLS Client is able to receive metadata flows in dynamic network environments by analysing the lateness of each link as they it is received. If no link has been received during certain period, the client issues an STIME command to ensure synchronisation with the server.

This initial TLS prototype stores metadata as simple unidirectional links on the server. In the following section, we describe an implementation where links are stored and communicated using the Fundamental Open Hypermedia Model.

FOHM IMPLEMENTATION

Our second prototype system was produced by mapping the TLTP protocol onto the performatives of the Southampton Framework for Agent Research [Mor00]. The SoFAR framework supports the authoring of multi-agent systems for distributed environments, and is being used for research into distributed multimedia information systems. The result, as described in [Cru01], is that we define two agents, a service agent and a client agent that represent the TLTP server and client respectively. The service agent requests temporal link

Command	Description
`HELLO`	The server responds with "TLTP/1.0" to indicate that this server is capable of conversing the TLTP protocol.
`BYE`	Invoking `BYE` tells the server to close the connection.
`SELECT <URI>`	This command is used to select the media stream that the TLS server server delivers links about. The server responds with `YES` or `NO` to tell the client if it knows of any metadata about the selected media stream.
`STIME`	The `STIME` command requests that the server return the value of the server's local clock. The result is used by the client to determine a suitable value for the `BIAS` command.
`BIAS <n>`	The `BIAS` command informs the server of the time difference between the server clock and the client clock.
`ENABLE`	Tells the server to start metadata delivery of the temporal linkbase. It is important to note that the XML document is not delivered immediately; rather each `<tlink>` element is delivered shortly before its deadline.
`DISABLE`	Tells the server to stop metadata delivery. If an XML document is in the process of delivery, the document is finished to ensure that well formed XML is produced.

Table 1: Summary of client to server TLTP commands

data from an agent that conforms to the Fundamental Open Hypermedia Model [Mil00]. In this system, the TLS server agent in effect becomes a client agent for a FOHM server agent. Link data is inserted into the FOHM server agent's knowledge base, and is extracted in temporal order.

Temporal link information is stored within the FOHM model by extending items within the knowledge base with *time LocSpecs*. The FOHM client agent receives links with their associated LocSpecs. In a non-live scenario, we can simply perform a single query to the FOHM server agent, requesting all link information about a particular media flow. The live scenario, however, requires a more demanding conversation between the two agents. The knowledge base of the FOHM server agent is continuously expanded, thus the FOHM client agent is required to periodically query for link data that lies within a specified time range. In a finely tuned system, the FOHM server delivers a block of link information just before it is to be delivered as a continuous link flow to a TLS client.

The SoFAR framework performs a match-making service between agents: a FOHM client queries the SoFAR registry for FOHM server agents. Thus when a FOHM server agent is started within the SoFAR environment, it needs to register its ability to serve FOHM linkbases with the SoFAR registry. During this process the server agent can restrict the nodes within the network to which it will advertise services, so limiting the possibility of overloading. The matching service of SoFAR also allows an agent to specify the particular media flow a server delivers. So a FOHM server would only be matched with FOHM clients that asked for the advertised media flow, allowing us to add extra agents dynamically for a particular media flow when demand for that flow is high.

In contrast to the first TLS prototype, we note two features gained from integrating the Temporal Linking Service with FOHM and the SoFAR framework: firstly, the FOHM model allows us to add a temporal dimension to existing FOHM metadata by way of the LocSpec mechanism; secondly, the match-making service of SoFAR gives the TLS client a service discovery ability. In the first prototype the TLS client needs to know exactly which metadata server to access, as it requires the internet address and port number of the metadata server. The SoFAR registry enables TLS Servers to advertise their services, so that the TLS clients can ask the SoFAR framework directly about media flows instead of servers.

A DISTRIBUTED FRAMEWORK FOR CONTINUOUS METADATA

The Temporal Linking Service and FOHM implementation demonstrate the usefulness of continuous metadata. In this section we will develop these ideas within a more general conceptual framework for the delivery of continuous metadata in a distributed environment (such as the Internet). The greatest strength of the WWW as a hypertext system is its highly distributed nature; we see a framework such as this as a first step to providing a corresponding architecture for multimedia data and associated metadata.

Mediadata and Metadata

As with the TLS, the framework will deal with mediadata and metadata, where a separate and distinct continuous metadata flow carries additional data about a corresponding mediadata flow. Although the framework caters for a more general scenario, we would normally expect the mediadata to take the form of a multimedia stream, such as audio or video, which can be characterised as a continually evolving flow of data—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.

It can be argued that what may be metadata in one case should be mediadata in another, and in many ways this is true. While a flow of MIDI information would be metadata for a raw audio mediadata flow in one case, in another there may be no audio stream and the MIDI might form a mediadata flow 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

Client Command	Server Response
`HELLO`	`TLTP/1.0`

`SELECT`
`http://example.com/data/`
`events/20001023-1.mpg`
	`YES`

`STIME`	
`BIAS -961755418994`	

`ENABLE`	`<?xml version="1.0" encoding="UTF-8"?>`
	`<!DOCTYPE TLINKBASE SYSTEM "tlinkbase.dtd">`
	`<TLINKBASE>`
	`<TLINK START="0" END="297" LABEL="Xanadu, Zigzag and Zepler"`
	`FROM="http://example.com/data/events/20001023-1.mpg"`
	`TO="http://example.com/data/slides/zigzag/sld001.htm"/>`
	`<TLINK START="297" END="356" LABEL="Ted Nelson"`
	`FROM="http://example.com/data/events/20001023-1.mpg"/>`
	`TO="http://example.com/data/slides/zigzag/sld002.htm"/>`
	`</TLINKBASE>`

`DISABLE`	
`BYE`	

Table 2: Example TLTP transcript

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 amount of associated metadata).

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

Firstly, we will 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.

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 derived 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. 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. 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, discussed later).

The *metadata source* is the point at which a continuous metadata flow enters the framework. This may be at the same point as the mediadata source or it may be distributed at a different point: for a live news feed a provider might construct a metadata flow of relevant links at the same broadcast point as the mediadata; while 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 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. If we presume that the mediadata flow will be a continuous stream, then in a recorded broadcast scenario pre-compiled metadata can either be sent along the appropriate metadata stream to arrive ahead of the relevant mediadata, or held back to be transmitted in near synchronisation with the mediastream. In the first case the receiving end of the metadata flow must buffer the data, 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 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. Of course, in a live two-way conversational situation the scope for delay and buffering is greatly reduced; otherwise the system would become unusable.

In all cases, the greater the coordination of the flow, the lesser 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* - although RTP could be used over a reliable transport its whole design is tailored to an unreliable scenario, hence is it unsuitable. (At best, a contrived transport could be designed to assign one atomic 'chunk' of metadata to each transport level packet so that if one packet were lost or out of order it would not invalidate other chunks; we do not see this as a very feasible solution). In the future we expect QoS enabled networks will allow guaranteed delivery of the type required to occur.

To encode the metadata flow above the transport layer, a derivative of the TLS XML markup would seem suitable, and if the metadata carried in its payload is also encoded using XML, the namespace [Bra99] mechanism could be usefully employed.

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.) 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 used at a presentation point must have been derived from that 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):

1. The metadata must have an identifying code. When multiple flows are combined at a presentation point an identifier is needed to deal with packets from a particular flow in a consistent manner; the identifier should also allow derivation of the mediadata flow with which it must be synchronised.
2. To synchronise the media and metadata, each packet of metadata must have a pair of validity timestamps bounding when the metadata is true in relation to the mediadata and the timing information embedded within it.
3. For user presentation there should be another pair of timestamps bounding a extension around the valid time, during which it is suggested that the metadata is displayed (although this could be overridden by user presentation preferences).
4. To present the metadata in a suitable manner there must be a code to describe the content type of the payload the metadata packet is carrying. Although the content code needs to be standardised within the framework, the format of the content itself need not be.

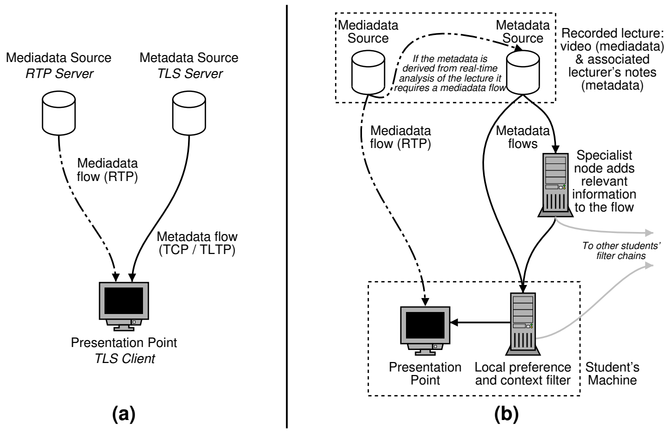

Figure 2: Simple framework configurations: (a) The Temporal Linking Service; (b) A unicast implementation of the recorded lecture presentation scenario

Once the mediadata and metadata have been delivered by the framework to a presentation point, it is expected that actual display of the information to a user would take place in cooperation with a mechanism suited to the purpose, e.g. a SMIL [Bug98] enabled browser.

At this point we can use the features of the framework to describe the TLS system developed in the previous section (figure 2(a)).

Filters

While the ability to select different metadata sources for a particular mediadata flow is useful, the real flexibility of the framework is through the introduction of processing nodes between the metadata source and the presentation point.

These filter nodes are distributed throughout the framework, taking one metadata flow as their input, modifying the metadata in some way, and then outputting 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 filter chains introduced in the Microcosm Open Hypermedia System [Dav92]. 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 de-multiplex 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.

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: to give the filters as much processing time as possible, metadata should leave the source as soon as it can, which would normally require mediadata buffering at the presentation point.

The end effect of a filter should be to either add or subtract metadata from that which a user receives 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 must 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. Suggested presentation relationships (flow x must be presented with flow y, but should not be presented with flow z) also need to be encoded in the metadata flow.

Control

Even with buffering at the presentation point, network congestion could delay metadata flows which need to be hard synchronised with others; in this situation the stalled flow can either be dropped, or the remaining flows must be paused while waiting for resumption. Other pauses or temporal movements may be user controlled, since the metadata flows must be paused once the mediadata is. To provide such functionality within the framework there must be control channels between the presentation point and the various filters and sources that feed it; these might use a suitable control protocol such as RTSP [Sch98].

There are two general approaches to propagating the control messages: Send the control message from the presentation point to the media and metadata sources, then propagate the message to the next filter in the chain; or send the control message to all the filters one hop "upstream" of the presentation point, then propagate the message up through the filter chains to the sources.

DEVELOPING THE FRAMEWORK FOR MULTICAST FLOWS

To fulfil the fourth requirement for distributed multimedia applications the framework must also support multicast connections. But the introduction of multicast capabilities also vastly increases the functionality and flexibility the framework can provide through its flows and filters.

Sources, Flows, and Presentation Points

Although media and metadata sources remain largely unchanged, they now transmit data across the framework using *multicast* flows. A source will send a particular media or metadata flow to any number of filter nodes or presentation points in a multicast group. This approach is most advantageous in a live broadcast or group presentation environment since all the nodes receiving a flow can receive the same (media or meta) data in the same temporal moment.

In the case of a single presentation point requesting a flow there is no 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.

Filters

To receive a metadata flow from another filter or source, a filter node only has to join the multicast group to which that flow is being sent. 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.

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.

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 on which the flow is being transmitted; a much more elegant solution than maintaining state about multiple point-to-point links.

Other facets of control inevitably become more complex with the introduction of group communication. A single source or filter can be expected to control flows to many other nodes, and any control messages to or from these nodes regarding flow must be dealt with, raising several important questions: How large a temporal discrepancy is needed for a source or filter to launch a new (time offset) flow of the same output, rather than relying on buffering within the framework? If one downstream node of many pauses its flow should the other flows continue whilst the paused flow is forked? Should the data be buffered for as long as resources allow before requesting another forked flow from the upstream node?

SCENARIOS

Having described the framework it is useful to present some motivating scenarios.

Live News Broadcast

A television news programme could be augmented by the broadcaster using metadata. The audio-video stream of the programme itself would be the primary mediadata flow, broadcast to many users with multicast. Classifications of prospective news items and clips might be conveyed in advance but as a broadcast is compiled on-the-fly so the metadata must also be generated while the broadcast occurs.

Subtitles, although traditionally embedded within the video stream, could be transmitted within a metadata flow and merged back with the mediadata at presentation points subscribed to the multicast group. Specialised filter nodes would also receive the flow via multicast and translate it into other languages which would in turn be multicast to presentation points requesting that particular language. In this way a single language broadcast can be viewed with subtitles in any language for which there is a translation node; each translating node can process multiple subtitle flows for different broadcasts.

Specialised metadata sources may provide compute intensive analysis of the video stream and create metadata. This is particularly valuable when the broadcast includes live material which does not have associated metadata, or when other material without appropriate metadata is used. Techniques include scene segmentation and face recognition. Metadata might also be generated based on existing metadata streams - for example, information about the sources of material might be expanded to information about permissions to reuse material.

Links to related material can be provided by the originating broadcaster and from other sources such as the image analysis node suggested above. Since they are transmitted as multicast metadata flows, a user can receive links both directly from the broadcaster and from nominated intermediate filters. These filters would take links from the original flow (optionally resolve the links) and multicast a new flow of links derived from a specialist linkbase. If more local to the presentation point, such a filter might provide a linkbase built upon a user's previous interests or current context. See figure 3.

Music Performance

While rehearsing, a musician might transmit a stream of their music to a specialist node (either remote or local in relation to the musician) 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 processed by a filter node which would analyse the melody and return a match from its database. Using a combination of the match from the database and the original MIDI metadata flow for tempo, a second filter node could use a remote music library to output a flow of additional MIDI data for the matched tune. This could provide suggestions for related music or even an accompaniment for the melody, which the rehearsing musician could receive and play along to. We have exercised this scenario through our tools for content-based navigation of music based on melodic pitch contours [Bla98a].

Lecture Presentation

A lecturer may annotate a class by providing links to the relevant point in the slides or online notes as a metadata stream. This could be used directly by students with presentation points in the same physical space as the lecturer, or accompanying a mediadata broadcast of the lecture for distance learning. Whatever the location of the presentation point, it would display the parts of the notes to coincide with that temporal space in the lecture.

The framework can also accommodate any branching in the presentation: to give further explanation of a particular subject the lecturer can refer back to an earlier point in the presentation or even to a previous lecture. The media and metadata streams will be reset accordingly, enabling the lecturer to include both live and pre-recorded material in the same class.

The lecturer might also suggest that students utilise a specialist linkbase for that particular subject area. The linkbase would interact with the framework though a filter node, receiving the metadata flow of the lecturers' 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 an extra linkbase 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 linkbase with the rest of the class, a mechanism the student could also use to distribute any insights or annotations they have added during the lecture. Once given, the entire lecture including metadata annotations can be stored for replay to both individuals and groups of students.

A unicast realisation of this scenario is shown in figure 2(b).

CONCLUSIONS

In this paper we have presented the case for continuous metadata, and proposed a framework with the following features:

1. Metadata is continuous and traverses the framework in a temporally significant manner.
2. The framework contains three types of node: sources, filters, and presentation points.
3. Mediadata is the flow against which other metadata flows are synchronised. It would normally be the temporal multimedia stream the metadata is derived from.
4. 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.

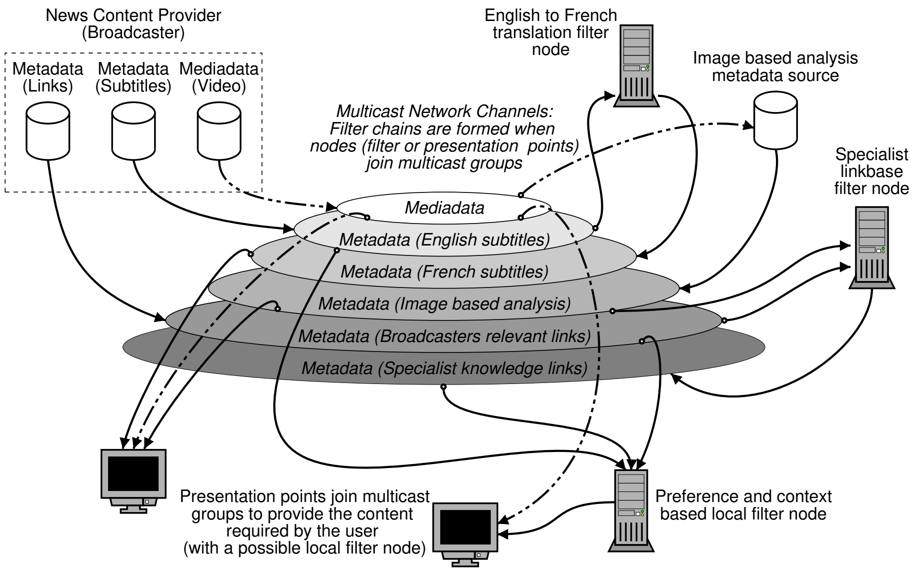

Figure 3: A framework configuration for the live news broadcast scenario

5. Transmission of media and metadata between nodes should be multicast where possible.
6. Metadata flows must be carried by a reliable transport.
7. The metadata carried within the flow (its payload) can be in any recognised metadata format, but must encode information to synchronise and present the payload data.
8. Filter nodes perform processing on an incoming metadata flow and output the results in another (amending the identifying and derivative codes 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.
9. Control channels between nodes must manage the rate of transmission and buffering of the temporal flows.

We have explored the requirement for streams of links to accompany multimedia streams, and for these to be synchronised when some part of the link delivery must be in real-time. Two experimental systems have illustrated these ideas, the second in particular demonstrating this work in the context of open hypermedia through use of the Fundamental Open Hypermedia Model (FOHM). Since we regard links as metadata, and we believe the linking scenarios extend naturally to other forms of synchronised metadata, our study makes a case for continuous metadata in general. We have presented a general distributed model for working with continuous metadata, and discussed the implications of extending this to multicast.

In the same way as open hypermedia promotes separable hyperstructure, we are promoting separable metadata. Many of the arguments in favour of open hypermedia are applicable here. At first sight it may appear that multimedia formats do not usually support embedded links, but we note that the emerging MPEG standards effectively promote embedded metadata and we anticipate that Web developers will instinctively embed URLs in media streams. In many applications there is a case for transporting digital media in a composite form including metadata, just as HTML with embedded links is an effective transport and delivery format. However, when we have multiple streams, described by one or more metadata flows, operating in a real-time scenario, there is a compelling case for handling the metadata separately.

Our future work is to realise the distributed metadata machinery using multicast and to evaluate the approach. The existing experimental systems will also be extended to further explore integration with other OHS components, particularly content-based navigation and application of ontologies. The distributed architecture raises systems infrastructure issues, especially with respect to timely delivery when working with mediadata and metadata streams together.

ACKNOWLEDGEMENTS

This research was supported by the UK Engineering and Physical Sciences Research Council through a studentship and is partially supported by project HyStream (GR/M84077, with Wendy Hall and Luc Moreau, in collaboration with BT Laboratories) and EQUATOR (GR/N15986). We are grateful to our colleagues in the IAM research group, especially Luc Moreau for his development of SoFAR, Dave Millard for insights and help with FOHM, and Neil Ridgway and others who have participated in our early prototype systems for streaming links.

REFERENCES

Adc93. P. Adcock, N. Davies, and G. Blair. "Supporting Continuous Media in Open Distributed Systems

Architectures". *Lecture Notes in Computer Science*, vol. 731:pp. 179–191, Oct. 1993.

Bla98a. S. G. Blackburn and D. C. De Roure. "A tool for content based navigation of music". In "Multimedia '98", pp. 361–368. ACM SIGMULTIMEDIA, 1998.

Bla98b. S. Blake, D. Black, M. Carlson, *et al.*. "An Architecture for Differentiated Services", Dec. 1998. URL `http://www.ietf.org/rfc/rfc2475.txt`.

Bou99. N. O. Bouvin and R. Schade. "Integrating Temporal Media and Open Hypermedia on the World Wide Web". In "WWW8", pp. 375–378. May 1999.

Bra94. R. Braden, D. Clark, and S. Shenker. "Integrated Services in the Internet Architecture: an Overview", Jun. 1994. URL `http://www.ietf.org/rfc/rfc1633.txt`.

Bra99. T. Bray, D. Hollander, and A. Layman. "Namespaces in XML", Jan. 1999. URL `http://www.w3.org/TR/REC-xml-names/`.

Bug98. S. Bugaj, D. Bulterman, B. Butterfield, *et al.*. "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", Jun. 1998. URL `http://www.w3.org/TR/REC-smil/`.

Cru01. D. Cruickshank, L. Moreau, and D. De Roure. "Architectural Design of a Multi-Agent System for Handling Metadata Streams". In "The Fifth Int'l Conf. on Autonomous Agents", May 2001.

Dav92. H. C. Davis, W. Hall, I. Heath, *et al.*. "Towards an Integrated Information Environment with Open Hypermedia Systems". In "Proceedings, ACM European Conference on Hypertext ECHT'92", pp. 181–190. ACM SIGLINK/SIGWEB, Nov. 1992.

Dav95. H. C. Davis. "To Embed or Not to Embed". *Communications of the ACM*, vol. 38(6):pp. 108–109, Aug. 1995.

Dee98. S. Deering and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification", Dec. 1998. URL `http://www.ietf.org/rfc/rfc2460.txt`.

DR98. D. De Roure, S. Blackburn, L. Oades, *et al.*. "Applying Open Hypermedia to Audio". In "Hypertext '98", pp. 285–286. ACM SIGLINK, 1998.

Geo96. N. D. Georganas, R. Steinmetz, and T. Nakagawa, eds. *Synchronization Issues in Multimedia Communications*, vol. 14 of *IEEE Journal on Selected Areas in Communications*. IEEE, 1996.

Goo95. S. Goose and W. Hall. "The Development of a Sound Viewer for an Open Hypermedia System". *The New Review of Hypermedia and Multimedia*, vol. 1:pp. 213–231, 1995.

Grø00. K. Grønbæk, L. Sloth, and N. O. Bouvin. "Open Hypermedia as User Controlled Meta Data for the Web". In "WWW9", pp. 554–566. May 2000.

Har94. L. Hardman, D. C. A. Bulterman, and G. v. Rossum. "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model". *Communications of the ACM*, vol. 37(2):pp. 50–62, Feb. 1994.

Mil00. D. Millard, L. Moreau, H. Davis, and S. Reich. "FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains". In "Hypertext 2000", pp. 93–102. ACM, May 2000.

Mor00. L. Moreau, N. Gibbins, D. De Roure, *et al.*. "SoFAR with DIM Agents: An Agent Framework for Distributed Information Management". In "The Fifth Int'l Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents", pp. 369–388. Apr. 2000.

Nac99. F. Nack and A. T. Lindsay. "Everything You Wanted to Know About MPEG-7". *IEEE Multimedia*, vol. 6(3):pp. 65–77, Sep. 1999.

Pan00. P. Pan and G. Davenport. "I-Views: A Community-oriented System for Sharing Streaming Video on the Internet". In "WWW9", pp. 567–582. May 2000.

Pim00. M. Pimental, G. Abowd, and Y. Ishiguro. "Linking by Interacting: A Paradigm for Authoring Hypertext and Hypermedia". In "Hypertext 2000", pp. 39–48. ACM, May 2000.

Rog00. B. Rogge, R. Van de Walle, I. L. Lemahieu, and W. R. Philips. "Introducing Streaming XML (SXML)". In "Proceedings of SPIE Vol. 4210", pp. 4210–4254. SPIE, Nov. 2000.

Sch96. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications", Jan. 1996. URL `http://www.ietf.org/rfc/rfc1889.txt`.

Sch98. H. Schulzrinne, A. Rao, and R. Lanphier. "Real Time
Streaming Protocol (RTSP)", Apr. 1998. URL `http://www.ietf.org/rfc/rfc2326.txt`.

Smi00. J. W. Smith, D. Stotts, and S.-U. Kum. "An Orthogonal Taxonomy for Hyperlink Anchor Generation in Video Streams Using OvalTine". In "Hypertext 2000", pp. 11–18. ACM, May 2000.