A QoS Adaptation Service for Mobile Streaming Applications[1]

Cristian Hesselman and Henk Eertink
Telematica Instituut
P.O. Box 589, 7500 AN, Enschede, the Netherlands
E-mail: {hesselman, eertink}@telin.nl

Abstract
Mobile multimedia applications typically operate in an environment that consists of a variety of different types of mobile hosts and wireless networks with different capabilities and resource availability. This heterogeneity makes it difficult to scale mobile multimedia applications up to large numbers of participants. In this paper, we propose a system that uses a limited set of domain-specific quality levels per service category to ensure scalability. The system combines IP multicast groups and proxies for that purpose. We furthermore introduce the mechanism that the system uses to adapt QoS levels as a result of host mobility. We discuss an implementation of this mechanism and give some qualitative results of its performance. Throughout this paper, we use an application that distributes a TV channel to mobile users to explain our work.

1 INTRODUCTION

Multimedia multiparty applications that integrate fixed and wireless communications typically run on a wide variety of hosts. As a result, the components that make up these applications usually have to rely on a wide variety of processing and communications capabilities [6, 11, 12, 35]. This is particularly true for application components that run on mobile hosts. Some of them for instance reside on high-end laptop computers with lots of processing power, high quality presentation capabilities and megabit network connectivity. Others reside on low-end PDAs with limited processing power, limited presentation capabilities and a low-speed network connection. This heterogeneous operating environment forms a problem for sessions that exchange audio-video streams between a large number of distributed application components (e.g. a session that distributes a TV channel). In such cases, it usually becomes unfeasible to deliver a stream to each receiving application component at a Quality of Service (QoS) level that is fine-tuned to the capabilities and the current resource availability (e.g. in terms of available network bandwidth) of the mobile host. An extreme solution to this problem is to deliver the same QoS to every application component in a session. This approach is relativly simple and desirable for some classes of applications, but it typically results in a large number of users receiving a suboptimal QoS.

The middleware[2] platform that we introduce in this paper strikes a balance between the above two extremes. This means that it supports sessions that can scale up to large

---

[1] Parts of this paper have also been presented at the 6th Open European Summerschool, Enschede, the Netherlands, September 2000, http://wwwtgs.cs.utwente.nl/eunice/summerschool/papers/paper4-4.pdf

[2] We view ‘middleware’ as a collection of generic distributed services that are application-independent.

numbers of application components, but that it typically does not deliver audio-video streams at fine-tuned QoS levels. In Section 2 we explain how our middleware achieves this. The basic idea is to divide the coverage area of a wireless infrastructure into domains and restrict the amount of available 'QoS spectrum' in each domain to a few QoS levels. In Section 3 we consider the mechanism that our platform uses to deal with host mobility. It boils down to transferring an application component to from one QoS level to another. We discuss an implementation of this mechanism and some qualitative results of its performance in Section 4. In Section 5 we describe related work. We present our conclusions in Section 6.

2 MULTIPARTY SESSIONS

Our middleware allows two or more distributed application components to exchange raw audio-video streams by means of a *session*. Consider for example an application that distributes a TV channel to mobile clients. Figure 1 shows the application components that are involved in the TV session.

The broadcaster's server component (S) produces a raw (i.e., uncoded) audio-video stream. The session delivers the stream to the player components P_1 through P_7 on mobile clients C_1 through C_7. Each player component P_i represents the media presentation resources (display and speakers) on client C_i and consumes the stream.

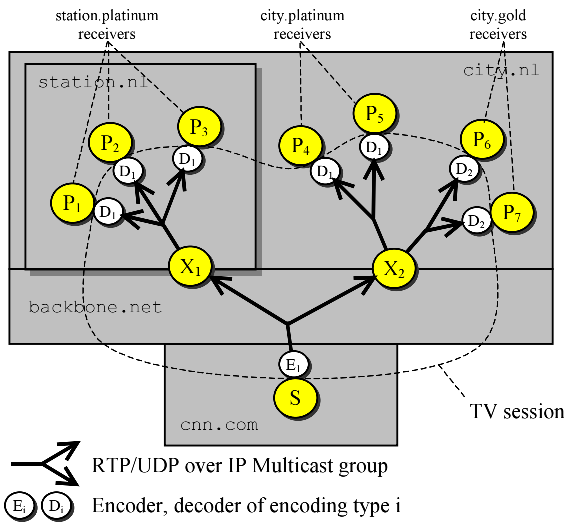

Figure 1. Multiparty TV Session.

The mobile clients are distributed over local area *domain* station.nl and over metropolitan area domain city.nl. They are equipped with media presentation resources (display and speakers), media processing resources (a decoder and an RTP [15] depacketizer) and with at least one wireless network interface. Clients C_1, C_2 and C_3 use the interface that connects to station.nl's high-speed wireless LAN. Clients C_4 through C_7 use the interface that connects to city.nl's medium speed wireless MAN. The common capability of the clients is that they are all IP multicast-enabled.

We follow [20] in the remainder of this paper. This means that our middleware encapsulates media processing resources (e.g. codecs, transcoders [10], and RTP (de)packetizers) and that it can rely on the availability of an end-to-end IP multicast connectivity service.

Service Classes

Our approach restricts the amount of 'QoS spectrum' available to the players in the TV session of Figure 1 to a few service classes. A *service class* defines a (perceptual) *QoS level* of the raw audio-video stream that a player receives. The capabilities and the current resource availability of the client on which a player runs largely determine the service class that it will get. To allow our system to scale sessions up to large numbers of players, we assume that the administrator of each domain defines a *small* number of service classes as well as the capabilities and available resources that a client machine needs to posses to receive each class. The price that we pay is that there will typically exist players that receive their audio-video stream at a QoS level that is not fine-tuned to the capabilities and the current resource availability of the mobile clients on which they run. Observe that we define *application level* service classes. This is unlike DiffServ [19] that defines IP-level service classes.

In the TV session of Figure 1, the players receive their stream at class platinum or at class gold. Player P_6 is for instance subject to a non-fine-tuned QoS if the traffic situation in C_6's cell is such there is enough bandwidth to attain a higher QoS level than that of class gold, but not enough to attain class platinum. Observe that the QoS level of the platinum service class of domain city.nl will generally differ from that of the platinum class in station.nl. To emphasize this, we denote these classes as city.platinum and station.platinum, respectively.

Realizing Service Classes

Our middleware defines the *QoS level* of a service class in terms of an audio and a video codec type (e.g. MPEG-4 [9]), a set of codec QoS characteristics, a packetizer type (RTP in this case) and a set of IP-level QoS characteristics. Figure 2 shows an example of what the QoS level of class station.platinum could look like.

The codec and IP QoS characteristics of class station.platinum predominantly determine the QoS level of the raw audio and video stream that the players associated with this class receive. To realize the platinum QoS level, codecs must be configured to `codecQoS` and IP must be configured to `ipQoS`. The latter can be accomplished through a QoS-aware IP-layer (e.g. an RSVP [19] enhanced one). Codec QoS characteristics are typically expressed in terms of audio sampling size, audio sampling rate, video sampling rate, and so on [17]. Typical IP-level QoS characteristics include minimum and maximum bandwidth, jitter, etc [17].

```
station.platinum = {
  codecQoS = {
    videoCodec = {
      type = "mpeg4";
      qosChars = {
        // platinum QoS characteristics
      }
    }
    audioCodec = {
      type = "mpeg4";
      qosChars = {
        // platinum QoS characteristics
      }
    }
  }
  packetizer = "RTP";
  ipQoS = {
    qosChars = {
      // platinum QoS characteristics
    }
  }
}
```

Figure 2. Service class station.platinum.

Players that receive the same service class form a relatively homogeneous group in terms of encoding types and codec and IP-level QoS characteristics. The clients on which these players run can therefore be interconnected directly. Our middleware uses one or more *site-local multicast groups* [21] for this purpose. For the sake of simplicity, we will realize a service class as one multicast group in this paper. In the example of Figure 1 this for instance means that clients C_1 through C_3 are interconnected by a single multicast group because players P_1, P_2 and P_3 receive the same service class.

Players that receive different service classes run on client machines that use different codecs or have significantly different codec or IP-level QoS characteristics. Our middleware uses *proxies* [4, 34] to bridge the differences between service classes. Proxies connect to a site-local multicast group for communications with mobile clients, and to a global multicast group to communicate with fixed clients. Proxies perform functions such as [13] rate adaptation, transcoding [10], audio and video filtering, and so on. For example, proxy X_2 in Figure 1 transcodes between MPEG-2 [8] and MPEG-4 [9] if the encoding of class station.gold is MPEG-4 (D_2) while that of the broadcasting server S is MPEG-2 (E_1). Proxies typically run on *gateway* hosts in an *access domain* such as station.nl.

3 MOBILITY

Host mobility typically causes resource availability to fluctuate [36]. Our middleware deals with this situation by transferring the player on the roaming host from one service class to another. We call this a *service class handoff.* A service class handoff effectively adapts the QoS of an audio-video stream that a player receives. As an example, consider client client C_3 of Figure 1 and assume that it roams from station.nl to city.nl. Figure 3 shows the portion of Figure 1 that is relevant for this particular situation.

As C_3 moves into the city.nl domain, the middleware will eventually need to hand off P_3 from class station.platinum to class city.platinum. To achieve this, the middleware unsubscribes C_3 from the multicast group associated with station.platinum and joins it to the multicast group that realizes city.platinum. This means that C_3 now receives the audio-video stream from X_2 rather than from X_1. The middleware also sets the parameters of the client's codec and IP service to the QoS level that city.platinum defines. In this particular example, the target platinum class is based on the same encoding type (D_1). The IP-level QoS definition of city.platinum will generally be lower because city.nl uses a relatively low-speed metropolitan area network rather than the high-speed local area network of station.nl. The QoS level of the raw audio-video stream that P_3 receives will therefore usually worsen when it roams to city.nl. As a result, the end-user perceives a degradation in the QoS of the audio-video stream that the player presents. The QoS of the audio-video stream usually improves when the client roams back to station.nl. Observe that inter-domain service class handoffs generally occur less frequently than network level handoffs that connect a host to a new base station.

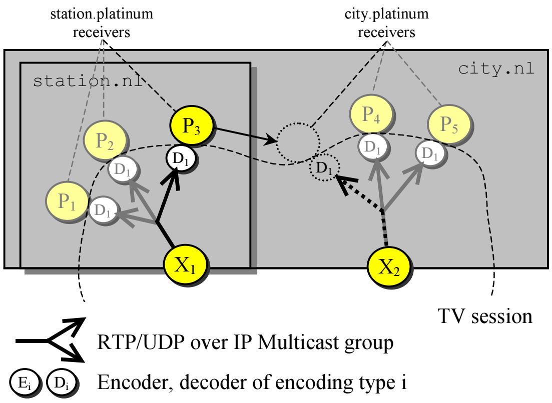

Figure 3. Inter-domain mobility.

There also exist *intra-domain* service class handoffs. In the example of Figure 3, this means that a player switches between different service classes while the client on which it runs roams within the same domain. Player P_3 may for example at some point need to switch from class platinum to class silver when C_3 roams within station.nl. The reason may be that C_3 roams from a lightly loaded cell to a more heavily loaded cell where there is not enough bandwidth to support class platinum.

The middleware *dynamically* creates a multicast group and a proxy if the target service class of a handoff is inactive. The handoff of Figure 3 for instance involves the creation of a multicast group and proxy X_2 if city.platinum is inactive (i.e., if C_4 and C_5 are not there).

4 IMPLEMENTATION

The goal of our implementation is to check if it is possible to handoff a player on a roaming client from one service class to another without serious hick ups in the QoS of the audio-video stream that it receives. In terms of the example of Figure 3, this means that we want to validate if it is possible to handoff a client from one multicast group to another. We specifically want to check if such a handoff is possible in an inter-domain mobility scenario. We therefore implemented the scenario of Section 3 in which client C_3 roams between domains station.nl and city.nl.

Testbed

Figure 4 shows the organization of our testbed. It also illustrates how the proxy and player components of Figure 3 are distributed over the machines in the testbed.

The Solaris machine hosts proxies X_1 and X_2. For reasons of simplicity, we have implemented proxies X_1 and X_2 to act as broadcasting servers. That is, they generate the stream containing the TV channel locally rather than from a stream coming from the broadcasting server S. X_1 and X_2 each consist of a QuickTime Darwin streaming server [22]: X_1 consists of server S_H; proxy X_2 consists of server S_L.

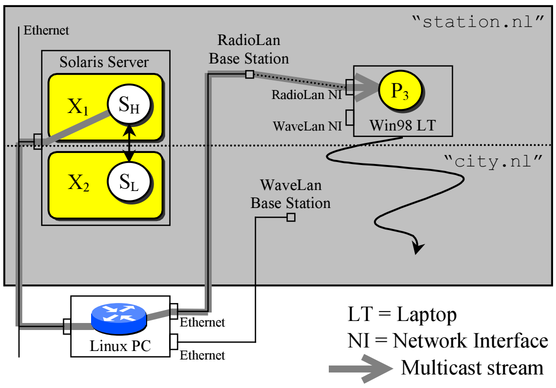

Figure 4. Testbed.

S_H and S_L run synchronously as indicated by the arrow between them in Figure 4 and loop continuously. S_H locally reads a high quality movie from a hinted (i.e., encoded) QuickTime file and transmits it onto the multicast group that represents class station.platinum. Similarly, S_L locally reads a low quality version of the same movie from a different hinted file and transmits it onto the multicast group that represents class city.platinum.

The Solaris server connects to a Linux PC through an Ethernet network. The Linux PC acts as a multicast router. It routes the traffic that it receives on its Ethernet network interface to one of two base stations. One base station uses a pre-802.11 version of Lucent's WaveLan [26] technology. It provides a gross over-the-air bandwidth of 1 Mbps. The WaveLan base station operates at a frequency of 2.4 GHz and has an indoor range of approximately 30 meters. The second base station in the testbed is based on a proprietary technology of RadioLan [27]. This base station offers a gross bandwidth of 10 Mbps. It operates in the 5.8 GHz band and has an indoor range of approximately 15 meters. The base stations are positioned such that the WaveLan cell *overlays* the RadioLan cell.

The two networks that we use mimic the local and metropolitan area networks of the station.nl and city.nl domains of Figure 3. The RadioLan network represents station.nl's local area network (high capacity, short range) whereas the WaveLan network mimics city.nl's metropolitan area network (medium capacity, medium range).

The multicast group that S_H uses represents the multicast group of class station.platinum. We have configured the multicast router such that it transmits the data that this multicast group carries onto the RadioLan network. This means that the IP-level QoS of this class has a best-effort QoS with a gross bandwidth of 10 Mbps. Similarly, the multicast group that S_L uses represents the multicast group of class city.platinum. The multicast router transmits the traffic of this multicast group onto the WaveLan network. This gives class city.platinum a best-effort IP-level QoS with a gross bandwidth of 1 Mbps.

A Windows98 laptop represents client C_3 of Figure 3. The laptop is equipped with a RadioLan and a WaveLan network interface and runs the QuickTime client software package [25]. We have built several control components around the QuickTime client package. One of these components is responsible for service class handoffs. This *handoff component* measures the packet loss characteristics on each network and hands the client off to the network that experiences the lowest loss. It thus implements a mobile-controlled handoff protocol [30].

As an example, assume that the laptop is located close to both the RadioLan and the WaveLan base station. In this case, both networks will hardly experience any loss. Our handoff component selects the high-speed RadioLan network in this situation. Other middleware components then initialize the necessary QuickTime components and join the laptop to the multicast group that S_H uses. As a result, the client receives the stream that S_H transmits over its RadioLan network interface and the end-user sees the high quality version of the movie. When the laptop gets out of range of the RadioLan network, it starts loosing packets. The handoff component detects this and informs the other middleware components. They reconfigure the QuickTime components, unsubscribe the laptop from the multicast group that S_H uses, and join it to the multicast group that S_L uses. As a result, the laptop now receives the low quality version of the movie that S_L transmits over its WaveLan interface and the user sees a degradation in the QoS of the movie. The middleware components go through this behavior in reverse order when the laptop roams back into range of the RadioLan network.

Qualitative Results

We have implemented the handoff component in several ways. One implementation measures the round-trip delay [35] between the client and the proxy to determine the most appropriate network to use. Another version measures the number of consecutive losses and hands the client off to another multicast group when a certain threshold is crossed [31, 32]. The implementation that provides the least *ping-pongs* [33] (assuming a sensible configuration) is based on common RF-level handoff mechanisms [29, 30, 33]. It is based on a sliding window with thresholds and hysteresis values, but on a packet level. We are currently gathering and interpreting the handoff delay figures of this last implementation. We will present the details of its operation as well as its performance in a forthcoming paper.

5 RELATED WORK

Proxies are a common way of dealing with capability variations of networks and hosts [1, 10, 13], in particular in a mobile environment [4, 12, 14, 33, 34].

Multiple multicast groups are a fairly common way of achieving scalability. Multiple multicast groups are for instance used to carry different layers of an encoded video stream [2, 3], for reliable multicast [16], for multicast flow control [23], and for large-scale distributed virtual environments [18]. Except for the latter, none of these approaches use multicast groups in a spatial fashion like we do.

We are also aware of work that combines proxies and multiple multicast groups [5, 6, 7, 10, 37]. This work mostly focuses on multimedia communications in a fixed environment. The exception is [6] that uses proxies and multicast groups to provide reliable communications to mobile hosts.

Work on handoff strategies and algorithms occurs at the RF-level [29, 30, 33] and at the packet level [28, 31, 32]. Our handoff component uses RF-like mechanisms at the packet level.

Finally, Mobile IP [24] supports host mobility in the Internet at the IP-level. Our platform, on the other hand, supports mobility at the middleware level.

6 CONCLUSIONS AND FUTURE WORK

We have proposed a system that trades optimal QoS delivery to individual application components for high scalability. The system revolves around the notions of sessions, dynamic application-level service classes, service class handoffs, site-local multicast groups and proxies. We have furthermore considered the mechanism that the system uses to adapt the QoS of an audio-video stream as a result of host mobility. We discussed an implementation of this mechanism as well as some qualitative results of its performance. We exemplified our work by means of an application that distributes a TV channel.

Our plans for the future are to extend our testbed, for instance with transcoding proxies, rate adaptation proxies and other types of wireless networks. We want to focus on the control aspects of our approach, for instance on the protocols that are required to assign a service class to an application component.

REFERENCES

[1] S. Cho, Y. Shin, "Multimedia Service Interworking over Heterogeneous Networking Environments", IEEE Network, March/April 1999

[2] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven Layered Multicast", Proc. of ACM SIGCOMM, Stanford, USA, August 1996

[3] X. Li, M. Ammar, S. Paul, "Video Multicast over the Internet", IEEE Network, March/April 1999

[4] E. Brewer, R. Katz, Y. Chawathe, S. Gribble, T. Hodes, G. Nguyen, M. Stemm, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. Padmanabhan, S. Seshan, "A Network Architecture for Heterogeneous Mobile Computing", IEEE Personal Communications Magazine, Oct. 1998

[5] E. Amir, S. McCanne, R. Katz, "An Active Service Framework and its Application to Real-time Multimedia Transcoding", Proc. of ACM SIGCOMM'98, Vancouver, Canada, Sept. 1998

[6] Y. Chawathe, S. Fink, S. McCanne, E. Brewer, "A Proxy Architecture for Reliable Multicast in Heterogeneous Environments", Proc. of ACM Multimedia'98, Bristol, UK, Sept. 1998

[7] K. Jonas, M. Kretschmer, J. Moedeker, "Get a KISS — Communication Infrastructure for Streaming Services in a Heterogeneous Environment", Proc. of ACM Multimedia'98, Bristol, UK, Sept. 1998

[8] T. Sikora, "MPEG-1 and MPEG-2 Digital Video Coding Standards", http://wwwam.hhi.de/mpeg-video/papers/sikora/mpeg1_2/mpeg1_2.htm

[9] R. Koenen, "MPEG-4 — Multimedia for Our Time", IEEE Spectrum, Feb. 1999

[10] E. Amir, S. McCanne, H. Zhang, "An Application Level Video Gateway", Proc. of ACM Multimedia, San Fransisco, USA, Nov. 1995

[11] A. Fasbender, F. Reichert, E. Geulen, J. Hjelm, T. Wierlemann, "Any Network, Any Terminal, Anywhere", IEEE Personal Communications, April 1999

[12] A. Fox, S. Gribble, E. Brewer, E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", ASPLOS-VII, Oct 1996

[13] N. Yeadon, A. Mauthe, F. Garcia, D. Hutchison, "QoS Filters: Addressing the Heterogeneity Gap", Proc. of the Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'96), Berlin, Germany, May 1996

[14] A. Balachandran, A. Campbell, M. Kounavis, "Active Filters: Delivering Scaled Media to Mobile Devices", 7th Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Louis, USA, May 1997

[15] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, Jan. 1996
[16] S. Kasera, J. Kurose, D. Towsley, "Scalable Reliable Multicast Using Multiple Multicast Groups", CMPSCI Technical Report TR 96-73, Oct. 1996
[17] "CORBAtelecoms: Telecommunications Domain Specification", OMG, June 1998
[18] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, P. Barham, "Exlpoiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments", IEEE Computer Graphics and Applications, Sept. 1995
[19] X. Xiao, L. Ni, "Internet QoS: A Big Picture", IEEE Network, March/April 1999
[20] C. Hesselman, I. Widya, A. van Halteren, L. Nieuwenhuis, "Middleware Support for Media Streaming Establishment Driven by User-oriented QoS Requirements", Proc. of the Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'00), Oct. 2000, Enschede, the Netherlands
[21] D. Lee, D. Lough, S. Midkiff, N. Davis, P. Benchoff, "The Next Generation of the Internet: Aspects of the Internet Protocol Version 6", IEEE Network, Jan/Feb 1998
[22] Darwin Streaming Server, http://www.publicsource.apple.com/projects/streaming/
[23] S. Bhattacharyya, J. Kurose, D. Towsley, R. Nagarajan, "Efficient Multicast Flow Control Using Multiple Multicast Groups", IEEE INFOCOM'98, San Francisco, USA, April 1998
[24] J. Solomon, "Mobile IP — The Internet Unplugged", Prentice Hall, 1998
[25] Apple QuickTime Client 4.1, http://developer.apple.com/quicktime/
[26] WaveLan homepage, http://www.wavelan.com
[27] RadioLan homepage, http://www.radiolan.com
[28] L. Taylor, R. Titmuss, C. Lebre, "The Challenges of Seamless Handover in Future Mobile Multimedia Networks", IEEE Personal Communications, April 1999
[29] G. Pollini, "Trends in Handover Design", IEEE Communications Surveys, http://www.comsoc.org/pubs/surveys/pollini/pollini-org.html
[30] N. Tripathi, J. Reed, H. VanLandingham, "Handoff in Cellular Systems", IEEE Personal Communications, Dec. 1998
[31] A. Seneviratne, B. Sarikaya, "Cellular Networks and Mobile Internet", Computer Communications, Sept. 1998
[32] M. Stemm, R. Katz, "Vertical Handoffs in Wireless Overlay Networks", ACM Mobile Networking, Special Issue on Mobile Networking and Internet, Spring 1998
[33] K. Pahlavan, P. Krishnamurthy, A. Hatami, M. Ylianttila, J. Makela, R. Pichna, J. Vallström, "Handoff in Hybrid Mobile Data Networks", IEEE Personal Communications, April 2000
[34] B. Zenel, D. Duchamp, "A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment", Proc. 3rd ACM/IEEE International Conference on Mobile Computing and Networking, Budapest, Hungary, Sept. 1997
[35] U. Varshney, R. Vetter, "Emerging Mobile and Wireless Networks", Communications of the ACM, June 2000
[36] A. Campbell, "A Research Agenda for QOS-aware Mobile Multimedia Middleware", NSF Wireless and Mobile Communications Workshop, Virginia, USA, March 1997
[37] I. Kouvelas, V. Hardman, J. Crowcroft, "Network Adaptive Continuous-Media Applications Through Self Organised Transcoding", Proc. of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98), July 1998, Cambridge, UK