Personalized Multicast

By

Guillaume Boissière

Diplôme d’Ingénieur
Institut National des Télécommunications, France, June 1996

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN MEDIA TECHNOLOGY
at the
Massachusetts Institute of Technology

June 1998

© Massachusetts Institute of Technology, 1998.
All rights reserved.

Signature of Author: ____________________________

Program in Media Arts and Sciences
May 8, 1998

Certified By: ________________________________
Dr. Andrew B. Lippman
Associate Director, MIT Media Laboratory

Accepted By: ________________________________
Dr. Stephen A. Benton
Chairperson, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

JUN 1 1998
Personalized Multicast

by

Guillaume Boissière

Submitted to the Program in Media Arts and Sciences on May 8, 1998
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Media Technology

ABSTRACT:

In the past few years, multicast protocols have been developed on the Internet to allow joint distribution of point-to-point and broadcast data. This begs the question of how one could bring television services to the network in a personalized, global and productive way.

This thesis demonstrates how live multicast Internet programs can be personalized by creating a recommendation system that suggests the most relevant programs to users. To make recommendations, this agent takes its information from several different sources such as what is the perceived intensity of the different programs available, what other people with the same interests as the viewer are watching, or what are the viewer’s interests and watching habits. A working prototype was built with Java to illustrate this model in the context of the Olympic Games with several meta-information multicast channels and locally stored video files. It also used the reliable multicast system with smart caching that was developed earlier at the Media Lab to receive multicast audio channels. Watching sports on the Internet with such an agent to help one sort which events are worth seeing would make the experience more enriching and less frustrating.

Thesis Supervisor: Andrew B. Lippman
Title: Associate Director, MIT Media Laboratory
Personalized Multicast

by

Guillaume Boissière

Thesis for the degree of Master of Science
at the
Massachusetts Institute of Technology

June 1998

The following people served as readers for this dissertation:

Thesis Reader: ___________________________ Dr. Pattie Maes
Associate Professor of Media Technology
Sony Corporation Career Development Professor of Media Arts and Sciences
MIT Media Laboratory

Thesis Reader: ___________________________ Dr. Michael Hawley
Assistant Professor of Media Arts and Sciences
Alex Dreyfoos Jr. (1954) Career Development Professor of Media Arts and Sciences
MIT Media Laboratory

Thesis Reader: ___________________________ Dr. Roger G. Kermode
Staff Engineer
Motorola Corporate Research Laboratory
Acknowledgements

Thanks to my parents and sister Lauriane, who were always there to encourage me all along my life. I would never have come to the MIT Media Lab without their love and support.

Thanks to my advisor Andy Lippman and who guided my work in the right direction during these two years. Andy’s critical and constructive thinking helped shape this thesis and taught me a lot. I must also thank him for letting me take a far-away trip to Asia and “Down Under” for a conference and sponsor meetings.

Many thanks also to my three thesis readers, Pattie Maes, Mike Hawley and Roger Kermode. Pattie gave me some very helpful advice about the interface and introduced me to the concept of “agents” in her class. Mike was a great motivator and gave me the opportunity to meet a number of people from the Olympic Committee and IBM that really helped strengthen this work. As for Woja, the guru of reliable scalable multicast with the Australian accent, his knowledge was a source of constant inspiration. He did a great job reviewing carefully this dissertation and pointing out its numerous weak points.

Thanks to the administrative staff of the Lab, who are really great, and particularly to Deborah Widener, Santina Tonelli, Laurie Ward and Linda Peterson.

Thanks to my officemates during these 2 years, Kathi Blocher, Jon Dakss, Ingeborg Endter, Chris Metcalfe, Doug Koen, Tom Slowe and Marko Turpeinen, who dealt on a daily basis with my French accent and somewhat European humor.
Thanks to many other people I have not thanked yet: Scott Smith, one of the best programmers I have ever met, who helped me a lot with HPRTP; Daniel Dreilinger, Bill Butera, Nitin Sawhney, Pascal Chesnais, Henry Holtzman, Martin Hadis, Joey Berkowski, Nuno Vasconcelos, Steve Waldman, Vadim Gerasimov, Brenden Mayer and John Watlington. You have all made my stay here more enjoyable and unforgettable.

Thanks also to many Media Lab’ers that I became friends with over the course of these two years, as well as to my friends from Ashdown House.

Last but not least, thanks to Gwelleh Hsu, who was always there when I needed her and whose humor and kindness helped me endure the tough Boston weather.
Table of Contents

ACKNOWLEDGEMENTS ........................................................................................................................... 7

TABLE OF CONTENTS ............................................................................................................................ 9

TABLE OF FIGURES ............................................................................................................................... 11

CHAPTER 1 INTRODUCTION ............................................................................................................. 13
  1.1 THE PROBLEM: THOUSANDS OF MEDIA STREAMS AVAILABLE ........................................... 13
  1.1.1 Transport ................................................................................................................................. 14
  1.1.2 Everyone Becomes a Publisher .............................................................................................. 14
  1.2 THE SOLUTION: AN AGENT SERVING AS A PERSONAL TELEVISION GUIDE .................. 15
  1.3 IMAGINE THE YEAR 2000 OLYMPICS .................................................................................. 16
  1.4 OVERVIEW OF THE THESIS ................................................................................................... 17

CHAPTER 2 MULTICASTING ON THE INTERNET ........................................................................ 19
  2.1 TRADITIONAL INTERNET PROTOCOLS .............................................................................. 19
  2.2 THE MBONE ............................................................................................................................. 21
  2.3 CURRENT ISSUES IN MULTICAST ....................................................................................... 21
  2.4 THE CANDY MODEL ............................................................................................................... 22

CHAPTER 3 A CRITICAL SURVEY OF TECHNIQUES TO PERSONALIZE INFORMATION IN THE DIGITAL AGE ................................. 25
  3.1 THE ART AND SCIENCE OF USER PROFILES ................................................................. 25
    3.1.1 Limitations of User Modeling ............................................................................................ 25
    3.1.2 Advantages of User Profiles ............................................................................................ 27
    3.1.3 Creating Dynamic Profiles ............................................................................................... 27
  3.2 SUMMARY OF FILTERING TECHNIQUES ......................................................................... 28
    3.2.1 Content-Based Filtering ..................................................................................................... 29
    3.2.2 Collaborative Filtering ....................................................................................................... 29
    3.2.3 User-Agent Relationship ................................................................................................... 29
  3.3 RATING POLICIES .................................................................................................................. 31
    3.3.1 Third-Party Ratings ............................................................................................................ 32
    3.3.2 Self-Rating .......................................................................................................................... 32
    3.3.3 Collaborative Ratings ......................................................................................................... 33
  3.4 BUILDING COMMUNITIES OF INTEREST ...................................................................... 33
    3.4.1 Established Communities .................................................................................................. 34
    3.4.2 Ephemeral Communities ................................................................................................... 35

CHAPTER 4 STRUCTURE OF THE MULTICAST OLYMPIC APPLICATION ................................ 39
  4.1 OVERVIEW OF THE APPLICATION ................................................................................... 39
  4.2 OVERVIEW OF THE AGENT’S SOURCES ......................................................................... 41
  4.3 THE “TELEVISION GUIDE” CHANNEL ............................................................................. 43
    4.3.1 Goal .................................................................................................................................. 43
    4.3.2 Implementation ................................................................................................................... 44
  4.4 THE “RECOMMENDATION” CHANNEL .......................................................................... 44
    4.4.1 Goal .................................................................................................................................. 44
    4.4.2 Implementation ................................................................................................................... 47
Table of Figures

Figure 1: Screen Shot of the Main Application Window ............................................. 17
Figure 2: Standard Internet Network Layers ........................................................... 20
Figure 3: Multi-Unicast vs. Multicast Delivery .......................................................... 21
Figure 4: Mechanism to select a user’s community .................................................. 35
Figure 5: Scalable Multicast Feedback for Ephemeral Communities ......................... 37
Figure 6: Overview of the Application ...................................................................... 39
Figure 7: Screen Shot of the “Recommendation” Channel Window ......................... 46
Figure 8: Screen Shot of the “Community” Channel Window .................................... 49
Figure 9: Screen Shot of the Main Application Window .......................................... 56
Figure 10: Module Relations in the Implementation of the System ......................... 57
Figure 11: Screen Shot of the Agent Policy Window ............................................... 58
Chapter 1  Introduction

The Internet is changing forever the way people communicate because it completely erases the geographical barrier between people and bits of information. Email has made it possible to receive in a few seconds electronic messages from all over the planet, at a cost that is independent from the location of both the sender and the recipient. On the World Wide Web, hyperlinks make it possible to jump from Brazil to Finland with a simple click.

Since the Media Lab’s creation in 1985, research has been undertaken in various fields ranging from holography to computer vision and music, to explore new ways of communicating and interacting in the Digital Age. The Digital Life consortium, created in 1997, “explores ever-present, personalized networking and the attachments to it that enrich our creative lives, from back-pocket PCs that listen to our stories when alone on the road, to tools for creating communities that span the globe. We abandon the uni-directional nature of home entertainment and allow the consumers of mass media to share in its creation.”

This vision of worldwide communities creating together is already a reality for chat groups on the Internet, but will undoubtedly extend to other mass media such as digital television. This thesis explores how these communities, along with information such as recommendation channels and personal profiles, can be put together to personalize live television events where there are multiple live channels such as the Olympic Games.

1.1  The Problem: Thousands of Media Streams Available

On today’s television, one can usually receive at most 200 channels, including satellite and cable television. And most American homes receive between 10 and 50
channels. The rapid growth of the Internet with an estimated 320 million Web pages available as of April 16\textsuperscript{th}, 1998 [Web98] will likely increase this number by several orders of magnitude for two reasons: transport and publishing.

\subsection*{1.1.1 Transport}

Today’s television is analog and uses airwaves to transport data from the television centers to people’s television sets. Because the frequency range reserved for television is limited and because the emitting frequency band of two different stations cannot overlap, there is only a small number of channels available at any location. This bandwidth problem could only be overcome either by allocating more bandwidth for television in general, or by finding ways to reduce the bandwidth to transmit the data for one channel. On the other hand, the bandwidth problem is also an issue on the Internet. Today’s standard modems, with a maximum transmission rate of 56 kb/s are not sufficient to transmit good quality video to people’s homes. Local Area Networks (LANs) using Ethernet or Token Ring make it possible to watch fair quality video stored somewhere else on the network, on condition than everyone else is not doing the same thing at the same time. However, recent advances in the domain of routing and transport [Cha97], such as work done on Gigabit Ethernet [Eng96] make it likely that bandwidth requirements will become less and less an issue in the future. And as soon as bandwidth is high enough to receive high-quality video on a home computer, the door will open to thousands of potential channels.

\subsection*{1.1.2 Everyone Becomes a Publisher}

Besides transport, the other reason why the number of channels will increase dramatically is publishing. The current proliferation of personal Web pages on the Web shows that with appropriate tools, people no longer hesitate to publish their own content on the Internet. With the increasing availability of cheap, simple digital audio and video
recording devices, thousands of audio and video channels made by users will appear, in addition to the millions of Web pages already available. This is what is referred as “Niche TV” in [Ker98], where each user can make his/her own television program and broadcast it on the network. Confronted with this new competition, traditional television networks will need a new generation of mass media applications that are more interactive and personalized, such as on-demand Internet television [Bry96].

In this wealth of live streams of information, one need to think about mechanisms that would help viewers locate and select which channels might be most interesting to them. In other words, they will need means to select and create personalized programs that they may be able to share with others.

1.2 The Solution: An Agent Serving as a Personal Television Guide

When large numbers of multicast video channels are present, scalability becomes an issue. The approach that was taken in this thesis was to built several separate multicast channels in addition to the regular media channels, that would serve several purposes:

- A “television guide” that announces and briefly describes the different programs available, as well as schedule changes,
- A “recommendation” channel that provides recommendations about the different programs available. These recommendations can be done beforehand by trusted third-party raters, such as the “Siskel & Ebert” recommendations channel for movies for example. They can also be done in real-time by the television networks or other entities for live events such as sports where the quality of the program is hard to predict in advance.
- A “community” channel that sends information about what other people in my community are currently watching.
From all these different sources of information, and also based on the user’s watching habits and known interests, the system is then able to propose a list of the most relevant channels to a user.

1.3 Imagine the Year 2000 Olympics...

“You are sitting in your couch watching the venues of several different Olympic events, when your personal agent pops up a window at the bottom of the screen saying: “Check out the women’s marathon”. You decide to look at what is going on with the marathon and change channels. A quick look at the statistics window indicates that a majority of people from your online community is already watching this program. Two marathon runners are about to enter the stadium, competing for a gold medal and are likely to break the world record. You stay on the channel and follow the last lap, conscious to witness a truly historic event.”

The interface described in this story may look like the one on Figure 1, where next to the video window, there is a list of the different channels available by supposed order of interest for the viewer. This interface, which was developed as part of this thesis, is described in more detail in Chapter 5.
Today, watching the Olympics or sports in general on television can be extremely frustrating. If a tennis game runs late, it is likely that the end will be cut to show the newscast instead, without any possibility to follow the end. And often the event one wants to follow is not shown because there is another event at the same time. Tomorrow, if we have thousands of channels available, the problem will be of another nature: which of these programs does the viewer really want to watch?

1.4 Overview of the Thesis

This thesis examines the issue of personalizing multicast and proposes a model to deal with real-time information in the context of television sport.
Chapter 2 describes the multicast backbone and examines the state of the art in multicast systems. The problems (scalability, reliability) that existing multicast protocols are facing are described and the current work is put in perspective. A overview of Roger Kermode’s work on multicast with smart network caches done at the MIT Media Laboratory completes this chapter.

Chapter 3 presents the existing techniques and approaches that have been taken to personalize and filter information before presenting it to a user. Four main fields of study are examined: user profiles, filtering techniques, ratings policies, and the building of communities of users.

Chapter 4 gives a general overview of the server in terms of design and implementation. Multicast scalable feedback schemes are examined to gather the information for the community channel. The three extra multicast channels used by the system (television guide, community and recommendation) to make its final recommendations are presented.

Chapter 5 focuses on the recommendation system that was developed for the Olympic server and describes how the agent uses the information it collected from the network to propose its recommendations to the viewer. It also addresses some questions such as advertising and graphical user interface.

Chapter 6, the concluding chapter, summarizes the findings and points out the new technologies and approaches developed as part of this work. Finally, new tracks of investigations and new scenarios are proposed.
Chapter 2  Multicasting on the Internet

2.1  Traditional Internet Protocols

The Internet began as a project sponsored by DARPA with the idea that big computers far apart could communicate together, and let users share processor time [Haf98]. More than 25 years later, the old Arpanet has become a worldwide Internet, but many of these pioneer protocols developed to let computers exchange messages and data are still used today.

Almost all of today’s Internet communications are based on the Internet Protocol (or IP) [Pos81], a network protocol that uses a simple packet format recognized by all the computers on the Internet. But since diverse applications have diverse requirements, several protocols can be added on top of IP to let network applications exchange information. TCP (Transmission Control Protocol) is a protocol designed for applications that require guaranteed delivery. Data can be lost in the intermediate network between two machines and TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received. This is the case for the HyperText Tranfer Protocol (HTTP) [BFF96] used by the World Wide Web. In contrast, UDP (User Datagram Protocol) is used by applications for which packet loss is not crucial, typically applications requiring real-time delivery such as videoconferencing. Figure 2 summarizes the relationships between these different Internet protocols.
All these protocols have historically been used in the context of communication between two parties. This type of communication is known as unicast; there is one sender and one receiver. Whereas unicast works very well for the World Wide Web where one client requests the information from one server, this does not extend well to applications where a server needs to disseminate or broadcast information to several clients simultaneously. One can think of multicasting as the Internet’s version of broadcasting. Data originates from one source, but it can reach everyone in the station’s signal area. The signal takes up some of the finite available bandwidth, and anyone who has the right equipment can tune in.

As shown in Figure 3, multicast gives a server the ability to “broadcast” information to potentially thousands or millions of users at the same time without creating a network congestion on the router. There is only one copy of the data sent per link, and the routing of packets to the hosts tuned on a multicast channel is left to the router, not to the server.
2.2 The MBONE

The MBONE, or Multicast backbone [Sav96], is the virtual network supporting multicast built on top of the Internet. It is a collection of Internet sites that supports IP multicast. Many of today’s Internet routers do not support multicasting. To transport the information through these routers and reach people worldwide through multicast, researchers have developed a technique called tunneling where the packets of multicast data are encapsulated into standard IP packets. These packets are sent across non-multicast capable sections of the network, and unwrapped at the other side. Since most of the recent routers on the market support multicast, one can expect that in a few years, the MBONE will cover a large part of the Internet.

2.3 Current Issues in Multicast

The two most pressing issues for multicast delivery are scalability and reliability. Multicast works well when there are dozens of receivers but it is difficult to scale to
thousands of receivers. However, applications such as digital television on the Internet would easily attract thousands or even millions of potential simultaneous viewers.

Another problem that people are trying to tackle is reliability [BrZ93]. Some applications, such as FTP or email, could let corporations reduce their internal network load by using multicast. Instead of overloading the network with hundreds of copies of the same message, this message could simply be multicast to the hundreds of receivers inside the corporation. Extensions of multicast FTP protocols are under way such as MFTP [MFT]. One can expect this type of new multicast applications to explode in the next few years, as the size of the digital messages exchanged between people gets bigger with the use of audio or video.

Recent works have tried to tackle these problems of reliability and scalability. This has given birth to a variety of protocols such as RMP (Reliable Multicast Protocol) [Whe95], RAMP (Reliable Adaptive Multicast Protocol) [Koi96], LGC (Local Group Concept) [Hof96] or SRM (Scalable Reliable Multicast) [Flo95] [Gem97].

### 2.4 The CANDY Model

In the Information and Entertainment section of the MIT Media Lab, Roger Kermode has shown that putting together a scalable reliable multicast transport protocol along with smart cache mechanisms could improve both scalability and reliability [Ker98]. This new model, called CANDY (Content and Application Negotiated Delivery) is based on two main ideas:

- There is no fundamental difference between senders and receivers. To the traditional client/server model, we can substitute a model where each entity has the capability to receive and retransmit media streams. This idea extends
previous work that was done in the Information and Entertainment section of the Media Lab with the Media Bank project [Lip96].

- Users can help other users get the information they need, therefore decreasing the required bandwidth on the network. A useful metaphor here is the metaphor of a classroom. Let's suppose a teacher is teaching a class, and one of the students, because he was finishing to copy something on his notebook, misses the last words the professor said. What is likely going to happen? Is the student going to interrupt the class to get the precious information? Although in certain cases he may, he will likely first ask his neighbor if he knows what the professor was saying, thus avoiding to interrupt the whole class. Now imagine that the professor is the server sending a live video stream over the Internet, and the students are the viewers. If one of the viewers misses a packet of data or wants to see the video with a short latency, instead of asking the server to retransmit it, he can make a query to see if any of his neighbors has the information needed. This prevents network traffic from happening outside the viewer's network neighborhood and therefore increases the scalability while adding reliability.

Not only this model provides an elegant way to reduce traffic through local scoping, but it also tends to create online communities of people who can share information and help each other.

After setting up in this chapter the multicast context of this thesis, the next chapter is dedicated to the different aspects of content personalization.
Chapter 3  A Critical Survey of Techniques to Personalize Information in the Digital Age

As Don Tapscott notes in his book “Growing Up Digital” [Tap97], the new generation of children that have grown up with a computer (he calls them N-Geners) keep asking for more interactivity and more personalization of their media environment. Personalization will be a key issue for a number of reasons. On the one side of the spectrum, simple personalization can help us be more effective by filtering email by order of priority based on some pre-defined criteria. On the other end of the spectrum, personalization can help us be more affective in our relationship with computers. As Roz Picard describes in her book “Affective Computing” [Pic97], if a computer knew enough about the user’s personality, it would be able to respond better to user’s frustration.

In this chapter, the most important techniques that are being used to personalize digital information are discussed and reviewed. These techniques fall into 4 main research domains: user profiles, filtering techniques, ratings policies, and the building of communities of interest.

3.1 The Art and Science of User Profiles

3.1.1 Limitations of User Modeling

Understanding how users’ interests can be modeled into personal digital profiles is the focus of research in the user modeling community. Their work has shown that building user profile is not a problem that admits a general solution. A good model for
one specific application may not work for an application that seems similar. Several reasons can explain this:

- Personality or tastes cannot be easily expressed through a list of keywords. "Easy-going" or "likes chocolates" may mean completely different things when employed for different people. This is a language limitation where what is said or written can take different nuances based on who is the speaker and what is the context.

- Culture is an integral part of someone’s personality, making nuances even harder to grasp. Understanding culture would require for a computer to have a huge common sense database that it could apply to different situations. Building such a computer with a huge database of common sense information is the main goal of Cyc [Len95], the most advanced project in this area that was started in 1984. But as of May 1998, Cyc is not yet able to read text by itself and create assumptions automatically. Understanding culture is then still far ahead.

- Assumptions inferred by knowledge-based systems based on profiles can prove wrong because such assumptions do not always work in real life. For example, if yellow is someone’s favorite color and hats are his/her favorite clothing accessories, it does not mean that he/she like yellow hats. Based on underlying cultural assumptions, he/she may think that yellow is too flashy a color for a hat.

- Research has shown that professional or personal new events in someone’s life create shifts of interest over time. Learning systems that monitor a user’s actions usually try to reach the ultimate user profile but do not consider that the user interests may have shifted. Lam et al. [Lam96] have shown that a system that detects shifts in interest can update a personal profile better than conventional systems.
3.1.2 Advantages of User Profiles

Even if user profiles are not the ideal solution to personalize digital media, they are an necessary start. First, they are easy to setup: putting a CGI script or a Java applet with a questionnaire on a Web page is the simplest way to get a sense of what people’s interests are. This is true for gathering simple demographics, or precise questions about a given topic. User profiles can give at best a broad idea about the person’s demographics and interests.

This information is well suited for most of the commerce and advertising growing on the Web. Knowing which segment of the population you belong to and which pages you have seen on the Web server is enough to show you a personalized advertising banner. The Open Profiling Standard or OPS [Hen97], developed jointly by Firefly, Verisign, Microsoft and Netscape, has the goal to enable “the trusted exchange of profile information between individuals and Web sites, with built in privacy safeguards.” Using OPS, a user would be able to setup his personal profile inside his Web browser. On request, a Web site would be able to extract the information it needs from the profile. Thus, the user would not have to fill out a demographics questionnaire each time.

3.1.3 Creating Dynamic Profiles

When it comes to personal agents that have to take decisions based on the users’ interests, personal profiles tend to become limited. One way to go around it is to create dynamic user profiles that learn over time what the user is interested in. This is usually done by monitoring the interaction of the user with the application, and letting the agent take decisions based on the usage of the system. Over time, learning algorithms make the profile evolve and stick more closely to the viewer’s habits. An example of generic user modeling system is Doppelgänger [Orw91], where information about users is provided by various sensors recording several aspects of the user’s computational
environment. The system is structured as a personalization server containing a number of user models evolving over time. Each user keeps control of the information other people can gather about him/her.

3.2 Summary of Filtering Techniques

On March 11, 1998, the ABC News Web site [ABC] started to display some form of personalization, by letting users specify the type of news they wanted to be displayed in the headlines Java applet of the server. The Amazon.com Web site [AMA] screens what people are reviewing and buying on their sites, using NetPerceptions technology [NET]. The next time they connect, they are advertised the books that they seem most likely to be interested in, based on their previous browsing on the site. My Yahoo!, using Firefly technology, let users specify their interests so that they would be only notified by types of news or stock changes they are interested in. Inquisit [INQ] lets users subscribe to news from dozens of sources delivered by e-mail according to keywords. In the domain of sports, SportingNews [SPO] lets you customize your news once for all. The first time you come to the site, you can fill a short questionnaire about what teams and sports you want to hear about, and which columnists you prefer.

Existing personalized services on the Internet rely on filtering techniques to sort information that might be useful to the user. Usually, this involves trying to sort or prune a large amount of data with some sort of predefined filter. Information filtering systems can be broadly divided into 2 distinct categories: content-based filtering and collaborative (or social) filtering.
3.2.1 Content-Based Filtering

The simplest and most used technique in this category is keyword matching. This is how query engines such as *Yahoo*, *Excite* or *AltaVista* return Web links based on a query. The idea is basically to lookup in the database and to return the occurrences that contain the most frequently the keywords entered. The Fishwrap news system developed by Chesnais et al. [Che95] at the Media Laboratory uses this type of algorithm, together with user models and the interaction with communities [Ben96], to propose a personalized Internet newspaper. Other techniques include semantic filtering, for example using associate network of keywords in a sentence [Rio95].

3.2.2 Collaborative Filtering

Another filtering technique commonly used is collaborative filtering, also called peer-to-peer or social filtering [Mal95]. Championed by *Firefly* and *NetPerceptions*, the idea is to use the recommendations of individuals that have similar tastes to figure out which piece of information is important. These systems are not parsing the documents at all but are trying to figure out if the current user would like it, based on what other users with similar interests thought of it. Such example of collaborative filtering systems include GroupLens [Res94], Ringo [Sha95] and Webhound [Las95].

3.2.3 User-Agent Relationship

Filtering is often associated with an agent that assists the user in a task. The agent, instead of filtering directly the information, recommends a decision based on its interaction with the user. In [Ter96], three types of communication between the user and the agent are described:
End-User Programming
The user describes the rules that the agent should use to process the information. For example, the user can specify "if new email priority = urgent → put on top of my email list". The main disadvantages of this approach is that (1) the user has to maintain the rules list over time, particularly when his/her interests change, and (2) the language to write the rules with has to be general and descriptive enough to adapt to new situations.

Programming by Demonstration
The user shows the system what to do in different situations and lets the system “record” the solution in its memory. The next time the user will be in a similar situation, the system will use this rule to propose a solution, such as demonstrated in the KIDSIM system [Smi94]. The shortcomings of this approach are that the system needs to be trained first by the user to be useful. Also, programming by demonstration works better when the graphical aspects of the applications are important, but not when it would require the agent to understand the meaning of a text for example.

Learning
The system is in monitoring mode, and tries to infer higher-level rules based on the user’s behavior with the system. The agent Letizia, developed by Henry Lieberman [Lie95] uses this approach. Letizia helps the user browse the Web by suggesting hyperlinks based on previous browsing. The advantage is that the user does not spend time programming the agent first, but the agent has also a slower learning curve and needs to be used intensively before being really useful. One solution is to use multi-agent systems that interact together and ask if an agent assisting another user has the solution to their problem. Another approach is to combine a learning algorithm with a few pre-programmed rules to help accelerate the learning curve.
Because learning algorithms provide a simple and long-term solution, the Olympic system detailed in Chapter 5 monitors usage to learn over time how the system is used, but also relies on a few user programmed rules.

It is interesting to note that programs that perform filtering for users can be associated with databases so that the history of the filtering can be kept. Looking at these databases with data mining techniques [Fay96] can provide some insights about how to handle a new request.

Before concluding on filtering techniques, it is also important to consider the user’s perception of the performance of a filtering system. Little work has been done in this domain but a study performed at Boeing [Fid97] suggested that users are not extremely sensitive to the quality of filtering. It also suggested that building and maintaining good user profiles were the keys to quality filtering in the technical context of the study.

### 3.3 Rating Policies

The filtering techniques described above often rely on content rating to perform their analysis. When querying documents with Yahoo for example, one is often presented documents that are considered the best of their category. These Web pages were reviewed by the Yahoo team, who highlighted the sites they considered the most relevant. There are three types of rating schemes commonly used on the Internet: self-rating, third party ratings and collaborative ratings.
3.3.1 Third-Party Ratings

The most common example of third-party ratings is being developed for parental control. With the opportunity for anyone to publish on the Web, it has become very easy to find inappropriate or indecent material for children. This situation has led several organizations to push for the development of rating standards that would help software bar the access to offensive or adult content to children. The principle of this technology is simple: a third-party company provides software that plugs into Web browsers and prevents from accessing inappropriate Web sites. It is up to the company to provide its own filters and lists of forbidden sites. More than thirty companies such as Cyber Patrol [CYB], NetNanny [NNA], or SafeSurf [SAF] provide this type of service.

In an effort to standardize ratings made by different companies, the World Wide Web Consortium has developed the PICS specification [Res96]. The goal of PICS is to provide a common description language for content incorporated as a header to a Web page. This language makes Web browsers aware of the rated content of the site before they even start downloading it. PICS encourages several third party ratings for a same page, so that people are free to choose the ratings service they want to use.

3.3.2 Self-Rating

The publisher of the document provides its own ratings or meta-information associated with the document. This could either use the PICS ratings or another means to add meta-information such as the Extensible Markup Language or XML [Bra98]. XML, a subset of SGML, was designed to let Web page designers create their own tags and give meaning to them. For example, people could have tags to reference a specific car or a field in a medical database. Combined with a simple Java parser [Bos97], XML is a powerful tool to add application-specific meta-information.
3.3.3 Collaborative Ratings

Instead of relying on a third-party authority, collaborative ratings are based on the fact that people belonging to the same online community tend to rate the content of a program the same way. Thus, when accessing the program, one can get some useful information about how people with supposed similar tastes appreciated it.

Collaborative ratings have the potential to be extremely useful but one of the issues is to provide incentive to propose ratings for others. The DJ Web Radio [DJW] lets users rate songs while listening, simply by clicking on a button. There is no other incentive to rate a song other than that of making the system better for others. This model can work if there are large numbers of people using the system or if there is a small group of dedicated users.

As one can see, collaborative ratings can be extremely powerful, but in an environment where there are a lot of users with different tastes, one needs to think about models for creating and maintaining these online communities.

3.4 Building Communities of Interest

The first question that comes up when we get a recommendation from someone is: what is my relation to this person and how does it make me trust his/her judgment? A careful look at this relationship is the answer to building successful online communities, where other members’ recommendations are really helpful.

Two models of online communities for collaborative filtering can be developed: 

*established communities* vs. *ephemeral communities*. 
3.4.1 Established Communities

The model of established communities supposes that people belonging to a community (1) are aware of it and (2) have created personal relationships with other members of the community. This is typically the case of some IRC chat channels, where “regulars” usually all know each other. This model does not mean that the community is static. New members can arrive at any time and existing members can leave, but there is a sense of belonging to a group of individuals, and the group is relatively stable.

Creating such communities can be long since people need time to get to know each other and create de-facto rules among the group. But because of the relationships between individuals in the group, recommendations from group members will be more trusted than recommendations from other sources.

One should also note that it is possible to artificially create established communities, by assigning a new user to a community automatically. By basing membership on demographics, one could create a community of females between 30 and 40 only for example. Another method for creating established communities is to gather data about the person and try to match him with one of the existing communities, either through email, friends, or any other way while respecting the person’s privacy.

A simple way to create stable communities for Internet television is to classify people based on their log on the system over time. If people have the same watching habits, they are more likely to see the same programs in the future. This implies that the viewers need to use the system for some time without being part of any explicit community, until enough usage data is gathered. In the case of multicast programs, one can imagine the following connection scenario that was implemented for the Olympic server described in this thesis:
On login, verify which community the user belongs to.

APPLICATION
(User belongs to C)

connects


Available Recommendation Channels

Figure 4: Mechanism to select a user’s community

In this case, the user only belongs to the community C, the “Tennis Fans” community. This has been determined over time as the user keeps watching tennis when there is a game going on. When connecting to the server, the user will automatically be directed to his “Tennis Fans” recommendation channel, where statistics and opinions from his “Tennis Fans” community are multicast in real-time about the different programs available.

3.4.2 Ephemeral Communities

The other model of community is ephemeral community. This would be very specific to live programs multicast on the Internet, since the notion would be to build on-the-fly communities of people with the same interests at the time of creation of the community. This type of communities would only last a few seconds, just enough to gather statistics about it.

The idea behind this model is the following can be illustrated by the following example. There are 3 live channels multicasting tennis, judo, and basketball on the
Internet and one wants the personal agents of different viewers to recommend them one of these programs. One can then ask random people connected to one of these three channels to send back the part of their personal profile where they have rated tennis, basketball and judo. Then on the server side, one can gather statistics and cluster the responses into three categories: what is the profile of the “average” tennis viewer, and the profile of the “average” basketball and judo viewers. Comparing these three profiles, the agent would then be able to decide which one is the closest to the one of the viewer, and make its recommendation based on that information.

The scheme to gather multicast feedback to create ephemeral communities is presented on Figure 5:
Since the requests from the server are continuously adjusted so to receive back approximately N responses from clients, there will never be a congestion of responses on the server side. The scheme proposed here is scalable then when the number of viewers increases. Ephemeral communities are composed of people whose agent recommendations are similar at a given time. Since agent recommendation are changing in different directions for different people, the communities such created are only a “snapshot” of an ephemeral situation.
Ephemeral communities are not used in the implementation of this work; established communities are used instead. The main reason is that people tend to trust people from their community more than an abstract community of people they don’t know at a given time. However, for future work, it might be interesting to try to use both and see which model works better.

After summarizing different approaches of personalization and their respective advantages, the next chapter will present the prototype application that was developed in the context of the Olympic Games for this thesis.
4.1 Overview of the Application

The application that was developed for this thesis is running on a PC with Windows NT 4.0 and the code is executed with the Symantec Java Virtual Machine. This application is organized in the following manner:

![Application Diagram]

*Figure 6: Overview of the Application*
Basically, the application is composed of a server entity, which contains a predefined timeline with sports programs. The server simulates the community and the recommendation channels behavior as if they were thousands of people connected, and multicast this generated real-time information on the Internet. The client side receives the information from the different channels and the agent proposes recommendations to the user based on the information it received.

The different media channels are represented here as A, B, C and D. Each of these channels is sent through a specific multicast channel on the Internet, on a separate set of IP addresses and port. In the working prototype, the video channels are played locally and the audio channels are multicast using HPRTP (Hierarchically Partitioned Reliable Transport Protocol) [Ker98] from another server running on a DEC Alpha machine on the network.

The central component of the Olympic server is the agent (or recommendation system) that rates the available live channels and updates for the user the list of what he may be most interested in. The recommendations of the agent are displayed on the side of the screen next to the video.

To make these recommendations, this agent uses a number of different sources, that can be easily set on or off from the interface. Basically, the user can control the source he wants to use to help the agent make its recommendations. These sources are respectively the “television guide” channel, the “recommendation” channel, the “community” channel, the personal profile and the usage of the system. All these sources, except the personal profile, which is static, evolve continuously. As the result, the agent has to update itself regularly (every 5 seconds in the current implementation) to take the new parameters of all these sources into account.
4.2 Overview of the Agent’s Sources

In Figure 6, there are three different types of multicast channels. Here are short descriptions of their roles:

➢ The “Television Guide” Channel
This multicast channel constantly sends information about the available programs. While the user downloads the list of available and future programs when he first connects to the Web site, this channel is essential because it announces to the users any program delays or extensions. This is particularly true for sport, where nobody knows when the program will end. It also announces last-minute new programs or cancelled programs, making the server much more dynamic.

➢ The “Recommendation” Channel(s)
This channel, also described as the “ratings” channel, gives the opportunity to the broadcasters or a third party to rate programs based on how they think they compare to other channels at the same time. The principle of this recommendation channel seems valid only in the case where there are independent parties grading different programs or if the source has no interest in being unfair. In a competitive environment with several grading services, these entities need to be reasonably fair or people will change grading service. Grading services may derive benefit from proposing recommendations to people. One way would be to give them the ability to push advertising on the user’s desktop.

➢ The “Community” Channel(s)
Many experiments in the domain of collaborative filtering have shown that one of the best indicators of what a person may want to watch is to examine what other people with the same interests would watch in the same situation
Creating communities of users sharing some common characteristics, and then rely on recommendations from people from these communities to select your program is the idea behind this community channel.

But people usually do not like to spend time rating programs, and have a more passive attitude. To take this into account, the community channel does not ask members of the community to rate channels, but simply gathers information about what members are watching. Therefore, the most watched channel is considered to be the most interesting for the community.

The User’s Personal Profile and his/her Usage of the System

Maybe more than for other types of programs, sports tend to attract always the same people for the same events. However, to make a good use of user profiles, several parameters need to be considered.

First of all, creating a user profile in the first place can be long and painful for the user if he/she needs to fill a 10 pages long questionnaire about his/her interests. The more precise the profile will be, the more likely the user will be tempted to go on another Web site, or only fill the mandatory fields of the questionnaire.

Second, people tend to describe their tastes and interests in a way that does not match exactly with what they would be really watching. One reason is the social constraints influencing our mindset: many people would not admit watching such programs because friends mock the program. Even if sport might be less sensitive to this type of deviation than movies or music, it needs to be taken into account. Another reason is the recent watching experience influencing our judgment: if the last baseball game one watched was great, then at the moment of filing the questionnaire, one tend to rate baseball higher than if the game was particularly uninteresting.

The approach that was taken for the implementation was to use both a static user profile describing how much the person liked different sports (graded
from 1 to 9) but also to monitor system usage and make statistics about what
the user watches over time. System usage is written to a log file and saved
for future uses.

4.3 The “Television Guide” Channel

4.3.1 Goal

Sport programs are among the most unpredictable type of programs one can think
of, both in length and in intensity. Who knows with precision beforehand when a
volleyball game will end? In traditional television, slots are reserved for the game,
based on how long the sport specialists think the game will last. But in practice, this
does not work well and when the game is longer than expected, it is not rare that the
channel decides to cut the end of the game to show the rest of its programs.

In the context of digital multicast channels, the constraint of the next program
cutting the present program does not exist anymore since there is an almost unlimited
number of available channels. Each program can have its own channel and start when it
was supposed to start, on condition the event is not delayed. We are here in a much
more dynamic environment, where one might not know until 10 seconds before the
actual end when the program will stop. To update users with this constant evolution of
available programs, one needs a “Television Guide” channel that would constantly sends
program announcements and correction information to the users.

There are two main types of messages that this channel needs to convey (1)
announcement and (2) deletion. In the following paragraph, the implementation of this
channel is detailed.
4.3.2 Implementation

The multicast IP address 224.15.7.5 associated with the port 6000 is reserved for the “Television Guide” channel. The packets sent over this channel follow the Sport Packet Description Format (SPDF), which is described in Appendix A.

The structure of SPDF is inspired by the structure of SDP (Session Description Protocol) [Han98], which provides a mechanism to announce general multicast sessions. Since the Olympic server application requires specific information, a special packet format was created, while trying to keep the advantages of SDP: (1) human-readable format, (2) portability and (3) simplicity.

For future versions, this format could be easily extended to include the description of the media types involved, or fields for time zones and encryption.

Several program descriptions can be sent in the same packet, by putting a separator between them. In the implementation, the server is sending a packet description for each of the 10 channels every second. On the client side, the list of channels is kept in memory and updated continuously.

For all of the meta-data multicast channels implemented in this project, only local tests inside the Media Laboratory have been performed, so the TTL (Time To Live) of all the packets was set at 4. This means that none of the packets was sent out of the Laboratory.

4.4 The “Recommendation” Channel

4.4.1 Goal

Any “real-world” model for digital television must take economic factors into consideration. There are two opposing models of revenues here. On one hand is the
Internet, which began as an ARPANET project aimed at connecting together several different university computers. Used strictly for research purposes, the Internet was historically not designed for commercial use. Even with the rapid growth of electronic commerce, there is still the notion that many free resources are accessible to the users and that everyone can publish its own. As the number of connected people increases, it becomes easier to find the same information through different sources.

On the other hand, the television model is quite different, since quality shows are expensive to produce. Material is more limited, and highly controlled since the revenues for the television networks are made out of the advertising that is put inside the programs. The movie industry follows a similar model: movies are expensive to make and cannot be freely given to the public. For these two worlds, the biggest fear is to have their video material freely circulating on the Internet without having the possibility for them to control who is paying to view it. The other pressing issue is to prevent from giving people the means to save and re-edit some of these materials.

In this context, giving the television networks the opportunities to rate their own programs may seem contradictory. They would obviously tend to rate higher the programs they have invested money in. How is this recommendation channel any useful then?

This recommendation channel may still be viable for three reasons:

1. In a context of competition between networks, we can imagine to have several recommendation channels competing with each other. For the Olympics, the images of the events are given to the television networks by an official Olympic entity responsible for putting the cameras at the venues. When a person uses the recommendation channel from a given network, this network would be the one to send the advertising. With such an incentive, and especially in the case where networks do not benefit directly from the program images like the Olympics, we
may expect recommendations to be still a bit biased but reasonably honest about the 
quality of the action.

2. Instead of having networks making direct recommendations about their programs, 
we could have trusted third parties grading the different programs available. These 
third parties could represent a community or an organization. As long as people 
trust the entity doing the grading, then there is no reason why they would not use this 
channel.

3. Maybe the most compelling reason why the recommendation channel is a key 
mechanism is the unexpected nature of sport competition. One of the intrinsic goals 
of the agent is to be able to detect extraordinary events going on that one may want 
to follow live. Otherwise, the system would not be better than today’s television, 
and one may miss the end of the most amazing marathon in history. Thus, there 
must be a mechanism for third parties (like members of communities or 
broadcasters) to rate the events for the viewers.

Figure 7: Screen Shot of the “Recommendation” Channel Window
4.4.2 Implementation

The ratings system that has been developed and implemented for the recommendation channel is simple: each channel can be rated on a scale ranging from 1 to 9. “1” stands for a program of minimal interest whereas “9” stands for an exceptional event that would occur. For sport, these ratings correspond to the level of intensity of the event.

The client can see the recommendation of the different channels available on a separate window that updates the ratings as soon as they have been received. The graphical user interface is presented on Figure 7. The currently available sport channels are displayed on the left with their corresponding ratings. The right panel displays graphically the corresponding ratings over time, letting the user follow the evolution of the ratings.

The reserved multicast address for this channel is also the address 224.15.7.5 but on port 6002. The structure of the packets that are sent over this channel is described in appendix B.

The simulation of the ratings for the different channels is very straightforward: for each channel, the simulator keeps track of the current value. Every 5 seconds, it calculates the new change in program intensity, equals to a gaussian of mean 0 and of standard deviation 1. It would be interesting to devise a model of how intensity varies over time for sport, but it seems there has not been any serious work done in this domain. Moreover, the goal here is more to create the mechanisms to make it possible to have ratings channels than to build a real-life model of how intensity in sport can be simulated precisely.

In the case where there are several recommendation channels, one would need to add a field describing who is the author of the recommendation. The other option is to reserve in advance channels for known entities doing program recommendations.
4.5 The “Community” Channel

4.5.1 Goal

As we have seen in Chapter 4, using online communities to feed the agent with information about what other users are watching can be extremely powerful. The more people from my community are watching a channel, the more likely this channel is going to be of some interest to me.

4.5.2 Implementation

The community channel that was implemented is using the IP address 224.15.7.5 together with the port 6001 to transmit this community information. As for the “recommendation” channel, the implementation encourages the possibility of having several community channels. If a user belongs to several communities, then averaging the different results would provide the final community information to the user.

As for the “Recommendation Channel”, a separate window opens with the community information, including the total estimated number of users from this community watching the Olympics on the Internet, as well as the respective audience shares for each of the current programs. The graphical user interface is presented in Figure 8 and evolves in real-time as new information is received.
On the right is displayed the total number of viewers from the community, with the audience shares for each of the programs available. The panel on the left displays these numbers in a graphical manner over time, so that the user can appreciate the evolution of the audience. Once a program has ended, it still appears in the left panel as "others" so that the user can have a precise estimate of the past total number of viewers.

In the case where there is a huge number of channels available, the current display would probably not work very well as the respective shares would all be small. In the case of the Olympics, though, there are no more than 16 events at the same time during the summer Olympics and 5 events during the winter Olympics [Ber98]. For such numbers, the proposed user interface still works pretty well.

The packet format used for the community channel is described in Appendix C. The simulator that is implemented for the channel is sending a packet every 5 seconds. This value was chosen arbitrarily and can probably be longer in general, since the audience shares do not change so quickly, even for sport. The simulator is simulating the audience on each channel as being independent variables, which is probably not exactly true for television. However, the goal of this thesis was not to built a precise
model of how audience works, but more to see how a fair audience model would work with a recommendation system.

For each channel, the simulator keeps track of the latest audience and the audience change (positive or negative) is defined as a gaussian with mean 0 and standard deviation equals to $1/4^{th}$ of the current audience.

As a side note, it is interesting to note that some of the biggest players in the world of the television and computers are working together on a way to estimate watching habits for digital media. On February 4, 1998, Nielsen Media Research and Microsoft announced they had developed the first metering system for digital Media that would track television viewing with the coming Microsoft Windows $98\text{TM}$ operating system.

### 4.5.3 Gathering Statistics About What my Community is Watching

Performing multicast feedback, i.e. determining what everyone is watching at a given time does not scale well. If millions of viewers are watching a program, then there is no means to query each of the machines and ask it to send back the program id of the channel the user is watching. However, we do not really need to know precisely what everyone is watching at a given time, but only what a sample of people are watching.

Several models [Bol94] [Shu96] have been proposed to provide scalable multicast feedback control in case we can use only partial feedback from the receivers. [ChA96] extends this scheme to a multiple channel scenario. Based on these ideas, the following simple scalable feedback scheme is proposed:
The server multicasts a query such as: all the clients whose IP address ends with 56 and belonging to community “Media Lab”, please send back the id of the program you are watching.

Based on the number of responses received, the server can determine how many people approximately are watching the program.

It can also adapt its next request so that it would receive the approximate number of responses he wishes. In the previous example, if the server only receives 10 responses but would like to get 50 next time, its next query may be “all the clients with IP addresses ending with 35, 36, 37, 38 and 39, please respond.”

Using the IP address to estimate precisely the number of viewers is probably not the best parameter to use because they are not exactly equally distributed (very few users will have an IP address ending by 0 or 1). However, we could perform simple arithmetic on the IP addresses to make the query more precise and equally distributed.

4.5.4 A Word on Security

There are some obvious privacy issues with such a multicast feedback system. The meta-data channels do not face such concerns since in general, the information they provide should be public and do not give away any private information. Security issues concerning HPRTP and the media channels are addressed in more detail in [Ker98]. User feedback is the key element of the system concerned with privacy. But obviously people would not want other people to know what they have been watching this afternoon. In the case of the Olympics, it probably would not matter so much to people to send publicly what channel they are watching. However, it would be important to make people aware of the fact that information about what they are watching is sent back to the server and give them the possibility to turn this option off.

There are different notions of privacy as defined in [Atk95]. Authentication is considered as “the property of knowing that the data received is the same as the data that
was sent and that the claimed sender is in fact the actual sender.” **Integrity** is defined as “the property of ensuring that data is transmitted from the source to destination without undetected alteration.” **Confidentiality** is “the property of communicating such as the intended recipients know what was being sent but unintended parties cannot determine what was sent.” Using public/private key cryptography such to encode the content of the messages sent back to the server (for example using the RSA [RSA78] algorithm) ensures that only the server can decode the message and know what the person is watching. An advantage of the proposed scheme is that the feedback system detailed here uses random call to clients. Responses from the clients can be very sparse and do not reflect their viewing habits. It is then hard to deduce from the packets sent back by the client how long it has been connected for example. In summary, most of the remaining privacy issues have more to do with the transport of the actual data on the media channels with HPRTTP or another multicast transport protocol than with the new meta-data channels introduced in this thesis.

### 4.6 The User Profile

#### 4.6.1 From Static ...

A user profile has also been setup for each viewer of the application. The user must fill a short questionnaire the first time he/she enters the server about his/her interests in sport. There are 28 sports on the program for the Sydney 2000 Olympic Games, of which seven sports have multiple disciplines.

For each of these sports, the user has to rate the sport on a scale going from 1 (really dislike watching this sport) to 9 (great fan of this sport). When a server announces a program, it sends information about the type of sport it is as specified in the SPDF and this information is later matched to the user profile.
4.6.2 ... to Dynamic

As seen in Chapter 3, using static user profiles is not enough so a usage-meter was developed for the application. Basically, a thread checks every second which channel the user is watching and writes in a local log file the sport that is on this channel. Future systems could include more information than just the sport such as country represented and famous players. By matching these keywords with the extra-information sent on the “Program Channel”, the agent would be able to detect if someone’s national team or favorite star is playing.

This usage monitoring is actually extremely important because over time, this information can be used to discover the viewers’ habits such as “how often in average does the viewer change channel in an hour” or “is there a correlation between the events that the user has followed till the end.”

The usage parameter is integrated separately from the user profile itself, so that it is possible to base the agent’s recommendations only on the usage and not the user profile.

The next chapter discusses how the agent uses all these different types of information to recommend the user which Olympic program to watch.
Chapter 5  Building a Recommendation Agent for the Olympics

5.1 Beyond the Personal Electronic Program Guide

The notion of having an agent that would help you figure out which digital television program to watch has been introduced in [Wit95] and [Ehr96]. They describe the personal Electronic Program Guide (EPG), which helps users filter information based on their interests. However, the described systems are different from the system proposed here. First, there are general-purpose systems where the media ranges from movies to shows or sport, so the content description is stays very general. The main difference with the system proposed in this dissertation is that these systems are only using personal profiles and program descriptions to filter the information.

Using online communities through multicast as well as specific recommendations from known entities adds a new dimension to the viewing experience. By contrast to television, which is by definition a solitary activity, we propose a model where users can exchange and share information, interact with each other through the programs and their communities. And by linking our system to the HPRTP protocol [Ker98], we can use smart caching to let users contribute with “their bits” to other’s experience.

5.2 Integrating User Actions, Profile, Communities and Recommendations

To understand better how the agent interacts with the viewer, here is a screen shot of the main application that was developed in Java [Gos96] using the Java Media Framework toolkit [JMF]. It should also be noted that since this application was built
with Java, it is possible to execute it inside a Web browser such as Netscape Communicator or Microsoft Explorer. The main window of the graphical user interface is presented below:

![Personalized Multicast: Using a Recommender Agent to help Users watch the](image)

**Figure 9: Screen Shot of the Main Application Window**

The central panel displays the video. For the simulation, 10 video clips in the MPEG format of approximately 2 minutes have been chosen. These clips are stored on the local machine, and played when the user selects them from the right panel. For future version, it would be interesting to multicast the video as well, but the Java toolkit that was used did not support multicasting of video with RTP yet.

The right panel comprises the list of programs available (4 here) listed by order of interest. The program at the top is the one that the agent would recommend with an overall grade of 660/1000. The program circled is the one that is currently being watched by the user, here the basketball.

The panel at the bottom displays fake advertising based on what the user is watching. A discussion about the role of advertising in digital television is included later in this chapter, as well as the motivation behind it here.
The different modules of the program are interacting the following way:

**Figure 10: Module Relations in the Implementation of the System**
Three history modules, represented on the top in gray keep information as it is being sent to them through the multicast channels. The program manager module keeps updating the list of the different program available. Every 5 seconds, it makes a request to the agent asking him to rate the different programs available, based on the new information it gathered. The agent returns this information to the program manager, which in turn updates the graphical user interface (GUI).

5.3 The Decision-Making Process of the Agent

From the information gathered from different sources, how does the agent make its recommendations?

The spirit of this project was really to propose to the user an agent that would be (1) helpful, (2) non-disturbing and (3) with a user-friendly interface. The non-disturbance aspect is particularly important and this is the reason why this agent does never change channels by itself or pop up a window where the user has to click. If the user is not interested in the agents' recommendations, they can be completely ignored. The same design principles were applied to the combination of different sources. Through a small panel shown below, the user has the opportunity to select the source(s) the agents should use to make its decisions.

![Agent Policy Window](image)

Figure 11: Screen Shot of the Agent Policy Window
The four possible sources are the one described in the precedent chapter: community, recommendations (also described as ratings on the interface), user profile and usage. Once the sources are selected, the recommendations are then simply based on the average of the values given for each of the sources. Since each of the sources return a grade between 0 and 1000, the result is sure to be also between 0 and 1000. Here is an example of this process:

<table>
<thead>
<tr>
<th></th>
<th>Community</th>
<th>Recommend.</th>
<th>User Profile</th>
<th>Usage</th>
<th>Final Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>689</td>
<td>938</td>
<td>367</td>
<td>344</td>
<td>(689+938+367+344)/4</td>
</tr>
<tr>
<td>B</td>
<td>345</td>
<td>266</td>
<td>347</td>
<td>836</td>
<td>(345+266+347+836)/4</td>
</tr>
</tbody>
</table>

Depending on the user, their relationship to their community, their faith in their ratings agency and their honesty about what they watch, some models are more accurate than others are.

In the long run, community and usage are the two most relevant parameters that are considered here, because they are both measuring the user’s behavior or relation’s to other’s behavior. Recommendations are also important to be able to be aware of unimportant unexpected events. Personal profile is more useful when the viewer starts using the system. Instead of having to bear an untrained system for a long time, it gives already some cues to the agent about what might be important. But after a repeated usage, it can probably be discarded.

### 5.4 User Interface and Advertising

When it comes to work on the user interface of this application several issues appear such as: should one display the different information available? Should the agent give a numeric grade to the programs or put words on them. Talking to people around the Lab and showing early demos of the work, it seemed that many of them were
Actually interested in being able to see the community channels and recommendation channels just out of curiosity. And these channels give actually very interesting pieces of information such as the evolution of the total number of viewers in your community or ratings comparison.

Another interface issue was to estimate what was the best way for the user to interact with the application in a seamless manner. Displaying information where the user has to click on moving objects is a complex problem and remains a well-known issue in hyperlink videos [Boi98a] [Boi98b]. In the prototype, the recommendations of the agent evolve at a reasonable pace; thus, it is rare that the user clicks on a region that he/she did not want to click on.

Another interesting side aspect of this project is the way the old models of television and Internet merge concerning advertising. Who is going likely to manage the advertising and present the advertising? Under which form? As a teaser in this prototype, a specific fake advertising was associated to each video clip based on the type of sport that was playing. This advertising was a banner such as the ones one can find on many Internet Web pages now. Even if the banner model is probably incomplete, one can imagine that two evolutions will probably take place. First, the traditional interruption of television programs will tend to be replaced with a permanent and less visible sort of advertising such as banners. Second, the content of advertising will be more and more targeted to the consumers that would actually be potential buyers for the product.

5.5 Effectively Using Smart Caching and Recovery

For audio only, the working prototype uses the multicast protocol HPRTP developed by Roger Kermode and ported to the Windows platform by Scott Smith. The HPRTP library offers a number of functions, which were used by this application:
• Query the network to figure out what is currently available
• Query the local cache to see what is stored there
• Start and stop the playing of a selected audio medium
• Setting the position in the audio stream

The smart caching potential of HPRTP is tremendous in a multi-agent environment. If a number of users have a personal agent that can communicate with other agents and negotiate bits of media they want to exchange, then we can imagine applications such as media agent agoras, where an agent would manage at best the viewer’s portfolio of media objects.
Chapter 6  Conclusion and Future Work

6.1 Contributions

The prototype that was developed as part of this thesis explores several new issues arising from the convergence of multicast, digital television, recommendation agents and sports. Here is a summary of the main contributions of this thesis:

♦ Implementation of the first real-time recommendation system for digital TV

By contrast to existing systems that make recommendations based on static information, this prototype is using dynamic, live multicast information to make its recommendations. As a result, the recommendations evolve over time even if the user is inactive, because the content itself evolves.

♦ Proposal of a realistic model for meta-information multicast channels

Taking into account the technical and commercial aspects, a model is proposed to add specialized multicast channels with meta-information. The three types of channels proposed here are the “television guide” channel, the “recommendation” channel and the “community” channel. A structure for each of these is proposed to deal with real-world situations, specifically sports in the prototype.

♦ Using live communities to get feedback

By contrast to existing works, this application uses information about what the communities of people with similar interests are watching. This unique metering system is full of promise in a digital world where online communities have become the Net equivalents of friends recommendations.
Scalable Feedback Model for Communities

A model to gather community information is proposed. Since we only need to know what a sample of people in the community are watching, we can make this model scalable by querying only a small number of people from this community and rotate the requests.

Integrating HPRTTP and other multicast channels in a graphical environment

Even if the use of the HPRTTP protocol was limited to audio channels multicasting music, the prototype shows how it is possible to integrate different types of media with different types of transport protocols.

Tools for building better programs

The working prototype was built as a set of tools that people can use at will to personalize their live program. Since it was written in Java, it should be easy to extend and port to different platforms. This toolkit is easily extensible to other sources of information that can help personalize the viewing experience.

6.2 Future Work

Multicast is still in its infancy and a number of problems (reliability, billing, security) need to be resolved before it can be deployed on a larger scale than the current IETF experiments. This work relies on multicast schemes for data delivery and as such, will not be extensible to a large scale as long as the underlying transport problems have not been resolved.

However, there are a number of outstanding issues that need to be addressed for future work. One interesting extension of this work would be to use these different channels in a distributed environment. The simulation that was created here uses only one server to deliver all the meta-data channels. Several problems arise if there are
several servers involved, for example if one server provides the community channel and another one the recommendation channel:

1. Security

Since the information is distributed, keeping track of who is sending what can be hard. A solution to this problem would be to authenticate the messages sent along the channels with a system of public/private key. This would solve the integrity and confidentiality problems [Atk95]. However, the anonymity problem would still remain and further work would be needed to find a workaround.

2. Synchronization

When different servers rely on the same initial real-time information, one need ways to communicate effectively between them. With transport latency and packet repair, one would need to provide mechanisms to ensure that different servers are reasonably synchronized, for example using the Network Time Protocol (NTP) [Mil92].

Another natural extension of this work would be to conduct real-life experiments with users to figure out what information is the most useful (recommendation channel, community channel, personal profile or usage). This would have to use real media and to be conducted over an extended period of time with several dozen people, otherwise it would be hard to judge adequately the usage and community channel’s added value.

Finally, it would be interesting to multicast video through HPRTP instead of playing it from the local disk in the current prototype. Given the amount of bandwidth necessary to multicast several channels of MPEG videos, there might be some bandwidth issues to solve before being able to use the same material as was used in this thesis. Another concern is that the Java toolkit that was used for displaying the video only accepts a local file or the HTTP protocol so far and does not yet provide the ability to play MPEG through real-time protocols such as RTP.

Nobody can really predict what the future of television on the Internet will be, but it is clear that the ubiquitous computers of the future will use more and more video
as it get easier and easier to handle. As the number of channels increases, people will need other people to help them sort out what is worth seeing from the background noise. As the importance of online communities grows, agents will become more and more the interface between people and their communities, since the real information, the one people trust and want, does not come from computers but from other people with similar mindsets.
References

Papers


Internet Drafts¹


World Wide Web Sites


¹ Internet Drafts are working documents that have a maximum lifetime of 6 months. If the draft becomes an RFC or authors do not revise the draft within 6 months of the date the current version was published, it ceases to exist. Internet drafts may be retrieved from the IETF Web site at http://www.ietf.org/
Appendix A: Structure of a “Television Guide” Channel Packet

The “Television Guide” channel packets have the following textual structure:

```
  v=<version>  : The type of packet (1 is announcement, 2 is deletion).
  t=<sport type>  : The type of sport.
  i=<id>  : The packet unique ID. This unique ID is defined as a string and must be created to reflect the timestamp of the beginning of the program.
  c=<channel>  : The IP address of the server multicasting on the video channel.
  o=<originator>  : The name and email of the originator of the data.
  n=<name>  : The name/title of the program.
```

```
v=1
t=ATH
i=98523879mar
c=18.85.1.57
o=Guillaume Boissiere <boissier@media.mit.edu>
n=Men’s Marathon
x=Ramirez is heading the race after 20 km.
x=Arrival should be around 5pm.
s=1980980809
d=98000
```
x=<extra info> : Extra information that may be relevant for the end user. Can be extended on several lines.

s=<start time> : Start time of the program, expressed in Unix time (in milliseconds).

d=<duration> : Expected duration of the program (in milliseconds). In case of sport programs where the end is not known in advance, new announcement packets can be issues regularly to update the duration.

The parser that was built does not require these fields to be order-specific
Appendix B: Structure of a “Recommendation” Channel Packet

The “Recommendation” channel packets have the following structure:

At 462346246
TheIdOfProgram1 7
TheIdOfProgram2 9
TheIdOfProgram3 4
...

At <time> : The keyword “At” followed by the time when the packet was sent. This is used if there is a packet loss to estimate at what time the recommendation was sent.

<id> <grade> : The program ID followed by the grade (from 1 to 9) for the corresponding program.
Appendix C: Structure of a “Community” Channel Packet

The packets sent over the multicast channel would typically look like:

```
3434
MyCommunityId
ProgramId1 43.23235
ProgramId2 23.4326
ProgramId3 12.453
ProgramId4 12.034
```

<total> : Total number of members of the community (estimated) currently watching one of the sport channels.

<community id> : String identifier for this community.

<program id><share> : For each program referenced by its unique id, the percentage of viewers on the system watching this particular program. The sum of these audience shares should be 100.