Intelligent Multicast Internet Radio

Goran Matic

A dissertation submitted to the University of Dublin in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

September 16. 2002

Declaration

I declare that the work described in this dissertation is, expect where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Goran Matic

Date: September 16, 2002

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Goran Matic

Date: September 16, 2002

Abstract

As the internet continues to mature static content will continue to give way to dynamic, interactive multimedia content. This content will enrich all online media while at the same time imposing heavy bandwidth usage on the underlying network infrastructure. Multicast networking will undoubtedly increase in popularity to relieve the congested networks.

Even as the scaling power of multicast networking approaches, users are burdened by the amount of multimedia and have to spend countless hours searching for the appropriate content. Recommender systems are a key way to personalize this content.

This thesis is aimed at developing a scalable streaming application over the current IP multicast infrastructure and then using personalization for recommending and customizing dynamic multimedia content.

The system uses a web based personalization system as the basis for user interaction and data collection. The multimedia streams are delivered through either Multicast or Unicast depending on each user's capabilities. In short it is a completely scalable, intelligent, audio streaming application.

Acknowledgments

This has been a truly enjoyable experience for many reasons. Not least of which was the friendships I've had the honor of making. Thank you for that, you know who you are.

Contents

1 Introduction.....12
1.1Motivation.....12
1.2Multicast Technology.....13
1.2.1Multicast Applications.....14
1.3Recommender Systems.....15
1.3.1Content Based Recommender Systems.....16
1.3.2Collaborative Recommender Systems.....16
1.3.3Content/Collaborative hybrid solution.....17
1.4Objectives.....17
1.5Document Structure.....17
2 Chapter 2 - State of the Art.....19
2.1Multicast Streaming.....19
2.1.1Multicast Transport Protocol.....20
2.1.2Push versus Pull Technology.....21
2.1.3Multicast IP Addressing.....21
2.1.4Mapping Multicast Address to MAC Addresses.....22
2.1.5Scoping Multicast Addresses.....23
2.1.6Multicast Transmission.....24
2.1.7Multicast Routing Protocols.....25
2.1.7.1Distance Vector.....25
2.1.7.2Link State.....26
2.1.7.3Link State Multicast.....27
2.1.7.4Distance Vector Multicast.....28
2.1.7.5Protocol Independent Multicast (PIM).....29
2.1.8Joining Multicast Groups.....30
2.1.8.1Internet Group Management Protocol (IGMP).....30
2.1.8.2Multicast Deployment.....32
2.2Multimedia Streaming.....33
2.3Multimedia Streaming Protocols.....34
2.3.0.1Multimedia Streaming Formats.....34
2.3.0.2Real-Time Streaming Protocols.....38

2.4Recommender Systems.....42
2.4.0.1The Cold Start Problem.....43
2.4.0.2Collaborative Based Filtering.....44
2.4.0.3Content Based Recommendations.....46
2.4.0.4Content based/collaborative based hybrid.....47
2.5Web Servers.....48
2.5.1Dynamic Web Content languages.....49
2.5.2Servlets.....50
2.6Summary.....50
3 Design and Analysis.....52
3.1User Tier.....53
3.1.1User and Website interaction.....53
3.1.2User/multimedia stream interaction.....54
3.2Web tier.....55
3.2.0.1SmartRadio Model (Mvc).....58
3.2.0.2SmartRadio View.....64
3.2.1SmartRadio Data Layer.....66
3.3The Streamer Tier.....67
3.4SmartRadio System Summary.....69
4 Implementation.....70
4.1User Side RTP Streaming Client.....70
4.2Multicast RTP Streamer.....71
4.2.0.1SmartRadio Icecast RTP Implementation.....74
4.2.0.2Icecast Protocol Implementation.....74
4.2.0.3Multicast RTP Transmission.....75
4.3SmartRadio Website Implementation.....77
4.3.1MP3FileLoader.....77
4.3.2SmartRadio Website.....78
4.3.2.1Presentation Layer.....78
4.3.2.2Application layer implementation.....79
4.3.2.3The persistence layer.....89
4.4Summary.....90
5 Evaluation.....91

5.1The user side.....91
5.1.1Stream client Application.....92
5.1.2Web client Interface.....93
5.2The Web Side.....94
5.2.1AI Recommender System.....95
5.2.1.1Suggestion System Enhancement.....95
5.2.1.2SmartPlaylist creation enhancements.....96
5.2.2Streamer Side.....96
5.3The Whole SmartRadio System.....98
5.4Summary.....98
6 Conclusion.....100
7 Bibliography.....102
8 Appendix A.....104
9 Appendix B: List of supported media types.....105

List of Figures

Figure 1 - Mbone Multicast Tunneling.....19
Figure 2 - Multicast TCP/IP Stack.....20
Figure 3 - Autonomous Systems.....24
Figure 4 - Multicast Routing Protocols.....27
Figure 5 - Multicast Group Membership.....30
Figure 6 - RTP Packet [RTP].....39
Figure 7 - TCP/IP Multicast Stack.....41
Figure 8 - Recommender system.....43
Figure 9 - Collaborative Based Filtering.....45
Figure 10 - Content Based Filtering.....47
Figure 11 - Content Based Filtering Example.....48
Figure 12 - SmartRadio Overall Architecture.....52
Figure 13 - E-Commerce Three Tier Model.....53
Figure 14 - User Tier Architecture.....55
Figure 15 - SmartRadio Web Tier.....55
Figure 16 - MVC model[SUN].....56
Figure 17 - SmartRadio MVC.....58
Figure 18 - User Clustering Formula.....60
Figure 19 - SmartRadio Suggestion Engine.....61
Figure 20 - Mean Squared Difference.....62
Figure 21 - Cascading Recommendation Engine.....63
Figure 22 - Stream Connector.....64
Figure 23 - SmartRadio Streamer.....68
Figure 24 - SmartRadio.....69
Figure 25 - Multicast Channel Transmission.....75
Figure 26 - Icecast RTP Multicast Streaming.....77
Figure 27 - Three Tier Model.....78
Figure 28 - Action Classes inheritance graph.....79
Figure 29 - SmartRadio Action Forms.....80
Figure 30 - StreamPool Class.....82
Figure 31 - SmartStream Events.....83

Figure 32 - SmartStream Class.....84
Figure 33 - SmartPlayList inheritance graph.....85
Figure 34 - SmartRadio Recommender System.....88
Figure 35 - Database Handler.....89
Figure 36 - Winamp Audio client.....92
Figure 37 - Zinf audio client.....92
Figure 38 - SmartRadio Web Page.....93
Figure 39 - Channel Registration Interface.....94
Figure 40 - User Clustering Formula.....95
Figure 41 - Icecast Configuration.....98

List of Tables

Table 1 - Scoped Multicast.....24
Table 2 - ID3 Tag Contents.....37
Table 3 - Icecast Protocol.....75
Table 4 - Adding to StreamPool.....81
Table 5 - Meta Refresh Tag.....82
Table 6 - Stream Event Model.....83
Table 7 - Timing Calculations.....85
Table 8 - SmartPlayList Loop.....86
Table 9 - List of Constants.....86
Table 10 - Recommender Pseudo Code.....89
Table 11 - Icecast console output.....97

1 Introduction

1.1 Motivation

Multimedia content, although generally extremely popular, has only recently experienced the internet revolution. It is the combination of the following three main factors that has spurred multimedia streaming to the forefront:

- The introduction of powerful and highly efficient compression algorithms such as the popular ISO-MPEG Audio Layer-3 algorithm.
- The increase in available processing power as indicated by Moore's law[1].
- The increase in available bandwidth such as cable, xDSL links for homes, and wireless networking standards such as 802.11 and Bluetooth for mobile devices.

These technological advances ultimately allow for download to play ratios of 1:1 or better.

Desktop and mobile clients, such as laptops or personal digital assistants (PDA), are no longer limited by local storage for multimedia content. Instead these same clients can connect to streaming servers and listen to or view broadcasts. This possibility, although exciting, is in fact very prohibitive for two major reasons. Firstly, clients must manually sift through seemingly endless choices of content which is not correctly indexed. This is known as the "*information overload*" problem. One solution to the information overload problem is the use of Recommender systems. Secondly, if too many members of the online community were to connect and start streaming multimedia content, the internet would quickly come to a grinding halt. This is because current inter-node *unicast* communication is not designed for streaming applications.

This thesis is aimed at first creating a scalable streaming application over the current IP multicast infrastructure, and then customizing dynamic multimedia content for transmission

1 Moore's law states that the number of transistors in an integrated circuit doubles every 18 months.

over this medium. This application is to be called *SmartRadio.*

To accomplish this objective, SmartRadio must be broken down into two inter-operable, but distinct parts:

- The Recommender system
- The multimedia streaming application

The recommender system discussed in this thesis uses personalization to tailor multimedia content for specific visitors. The main objective of the recommender system is twofold

1. Accurately predict and suggest multimedia content for individual users
2. Dynamic playlist customization

In order to meet these requirements, the recommender system uses a hybrid approach of both collaborative and content based filtering explained beneath. Finally the recommender system feeds streams to the multimedia streaming application which finally delivers it to the user.

The multimedia streaming application is responsible for delivering dynamic multimedia content to the end user. The main objective of this application is scalability. Scalability means that as the system grows over time, its capacity to service users does not diminish. This is particularly important in this application because of the inherent strain imposed upon network resources by multimedia content. Ideally, a multimedia streaming application such as the one suggested in this thesis would function over multicast IP instead of unicast IP. However, multicast IP has not yet gained widespread use and therefore another objective of the system would be to maintain backward compatibility with existing unicast multimedia clients.

1.2 Multicast Technology

In October 1996, the IP Multicast Initiative was formed to educate the user and networking communities about the benefits of multicast. The IPMI now encompasses more than eighty organizations worldwide. In early 1997, Microsoft, Intel and Cisco formed the Networked Multimedia Connection to promote multimedia and multicast networking. These internet giants are encouraging others and preparing themselves for the benefits of multicast

technology.

Multicast is a form of communication designed for one-to-many or many-to-many communication. This type of communication creates only one connection regardless of the number of users. Unlike traditional Internet traffic which requires a separate copy of the message for each source-destination pair, IP multicasting allows the source to transmit only a single copy of the message, which is then delivered to multiple destinations. Multicast-capable routers are responsible for forwarding the transmission to all networks hosting a user who is a member of the same group.

The bandwidth savings multicast provides are more dramatic when the number of open connections becomes large. Unicast bandwidth usage grows linearly with the amount of simultaneous connections open while multicast remains static. For example, using unicast streaming, a T1 connection running at 1.544Mb/s serving a CD quality stream of 128Kb/s would exceed its bandwidth at 12 users.

CD Quality Stream	***T1 Bandwidth***	***Theoretical Limit***
$(\frac{128000}{8}) = 16000\ Bytes/sec$	$(\frac{1544000}{8}) = 193000\ Bytes/sec$	$(\frac{193000}{16000}) = 12$

1.2.1 Multicast Applications

Multicast applications can be classed into two general categories:

- Real-Time Streaming Applications
- Non Real-Time Applications

Real-Time applications can function as either many-to-many or one-to-many. In many-to-many applications all users have the right to stream data in the same group. An example of a many-to-many multicast application is audio and video conferencing.

With one-to-many applications, one sender is identified as the transmitter and all others are receivers. An example of a one-to-many application is a real-time television broadcast. Real-time applications do not require absolute reliability, but do require precise timing without which they are rendered useless.

SmartRadio is a real-time application that has precise timing requirements. A transport

protocol that guarantees timing and reduces jitter must be used to stream over multicast.

1.3 Recommender Systems

As stated above, Recommender Systems use personalization to tailor content for specific visitors. Based upon a users past preferences a recommender system suggests new content. Specifically Recommender Systems base their decisions on comprehensive knowledge of who customers are, how they behave, and how similar they are to other customers. In essence, a recommender system learns from a customer over time and recommends products that they will find most valuable from among what is available.

Recommender systems are used by E-Commerce sites to suggest products to their customers. The products can be recommended based on:

- The Top overall sellers on a site.
- The demographics of the customer.
- The past buying behavior of the customer as a prediction for future buying behavior.

These systems however, are not new. Many businesses use Recommender Systems and their variants to customize content and to tailor each user's experience. In their book section, Amazon.com uses Recommender Systems in four different areas:

- **Customers who bought [SKR]**: Recommends content to users based on other similar users past purchases.
- **Eyes [SKR]**: Notifies users by email of new items added to Amazon.com.
- **Amazon.com Delivers [SKR]:** Periodically, editors at Amazon.com send email announcements to notify subscribers of their latest recommendations.
- **Book Matcher [SKR]:** Allows customers to give direct feedback on assets for consumption by other customers.

Aside from Amazon.com, CDNOW, eBay, Levis, Moviefinder.com, Reel.com and many others use Recommender Systems to enhance their customer's experience and to aid their customers in coping with the *information overload* problem.

1.3.1 Content Based Recommender Systems

Content based filtering provides recommendations by matching customers' interests with product or asset attributes[CKLT]. All products are described by a common set of features extracted from the available product descriptions. Individual customer's preferences are predicted solely from the products that he or she has rated, combined with the corresponding product features.

Content Based Recommender Systems have difficulties with synonymy[2], polysemy[3], and context [RISBG 1994] when judging the relevance of text. This is because computers cannot judge the true meaning of words.

1.3.2 Collaborative Recommender Systems

Collaborative based systems on the other hand, utilize the overlap of preference ratings among customers for product recommendation[CKLT]. If a user/asset preference rating is missing, collaborative Recommender Systems generate a recommendation based on the users similarity to other users.

This system does not suffer from the same problems as content based Recommender Systems because human readers do not share the computer's difficulties with synonymy, polysemy, and context when judging the relevance of assets. Moreover, people can judge assets on other dimensions such as quality, authoritativeness, or respectfulness. However, collaborative Recommender Systems fail when the system is first initialized or when a new user joins for whom there are no preference ratings.

1.3.3 Content/Collaborative hybrid solution

In order to circumvent the problems presented by collaborative and content based filtering, SmartRadio uses a hybrid approach. The hybrid approach uses the advantages of both systems and merges them into one solution, reducing the impact of the problems when each is used individually.

2 Synonymy means, equivalence of meaning
3 Polysemy is something that is characterized by many meanings.

1.4 Objectives

This section is divided into two categories: **Completed Objectives** and **Future work**.

Completed Objectives

1 Design of a scalable multicast streaming audio server that takes into account scalability and efficiency of current streaming protocols. The design should be compatible with both current audio streaming protocols and current multicast/unicast audio clients.

2 Implementation of this streaming server, taking into account system performance.

3 Design a fully functional online recommender system to intelligently customize dynamic content for consumption by the multicast multimedia streamer. The customization process should create dynamic playlists for groups of users. Furthermore the system should recommend appropriate streams or channels. Finally the system should remember all assets rated by users and infer ratings for assets a user has listened to but not rated.

4 Implement online recommender system.

Future Work

1 Testing and optimization of recommender system in a real time environment to tweak asset suggestions.

1.5 Document Structure

This section outlines the structure of the thesis.

Chapter 1: Introduction Describes the motivation behind the SmartRadio project and makes an introduction to the two major concepts involved:

- Multicast Push Streaming technology.
- Recommender System

The introduction also states the objectives.

Chapter 2: State of The Art Looks at the relevant material concerning Multicast networking and also relevant material in the area of streaming protocols and formats running over Multicast Technology. Also this section will look at relevant material concerning Recommender Systems and platforms for their development.

Chapter 3: Analysis and Design introduces the ideas behind Multicast streaming and its properties. This chapter also outlines the infrastructure over which these two applications will inter-operate. Additionally, this chapter introduces the ideas behind the recommender system and its properties.

Chapter 4: Implementation outlines the way in which the design decisions will be implemented and the decisions made during implementation.

Chapter 5: Evaluation examines the performance of the streaming solution and outlines future work.

Chapter 6: Conclusion Concludes the project.

2 State of the Art

This chapter describes the major underlying technologies used the components of SmartRadio. It also examines the technologies currently available so that an objective comparison can be made.

2.1 Multicast Streaming

Multicast networking and applications in the TCP/IP environment are not new. Proposals such as those made by Steve Deering and Dave Cheriton RFC[4] 966[DC 1985] were made as far back as 1985. Even though the benefits of large scale real-time or non real-time streaming over IP multicast is clear, it was not until the creation of the Mbone in 1992 that activity and interest in multicast IP grew significantly.

The Mbone provides a test bed for researchers and a vehicle to try out new multicast applications. It comprises a set of multicast-enabled subnetworks tied together by "tunnels" as shown in figure 2-1.

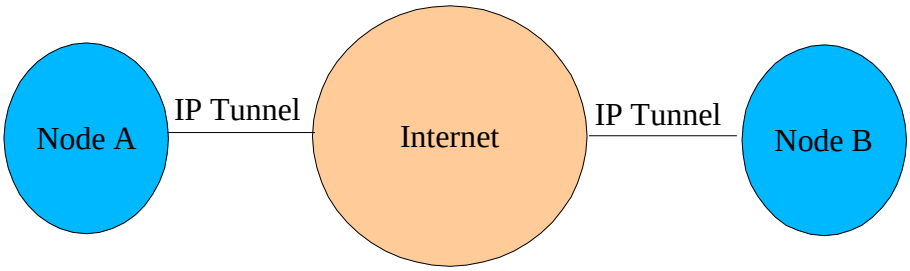


Figure 1 - Mbone Multicast Tunneling

The tunnels encapsulate multicast datagrams within normal unicast datagrams and allow multicast traffic to traverse parts of the network that do not support multicast natively. Since 1992, the Mbone has grown substantially. In March 1997, it included 3,400

4 Request For Comments (RFC) are the documents within the Internet Engineering Task Force (IETF) used to describe Internet specifications and provide other documentation. Internet Drafts are IETF documents that represent work in progress.

multicast-enabled subnets tied together [K 1999]. Organizations may request Mbone connections from their local Internet service providers. These connections may be used to watch an IETF meeting or share a whiteboard conference with members of another organizations. Many multicast tools have been created by university researchers[Mbone], and are generally available free of charge to the public

2.1.1 Multicast Transport Protocol

TCP/IP has two Transport layers that applications may use: The Transport Control Protocol often abbreviated to TCP and the User Datagram Protocol abbreviated to UDP. TCP provides applications with a high level of service. UDP on the other hand, provides only minimal services such as port multiplexing and error detection. If a packet is detected to be erroneous under UDP, it is simply discarded. Thus, out of order delivery, and missing packets are possible under UDP. Furthermore, TCP implements an end-to-end connection oriented abstraction over IPs connectionless datagram delivery service. UDP implements no such system.

However, due to the connection oriented abstraction provided with TCP, only unicast point-to-point connections are possible. Thus as show in figure 1-1, Multicast must run over UDP.

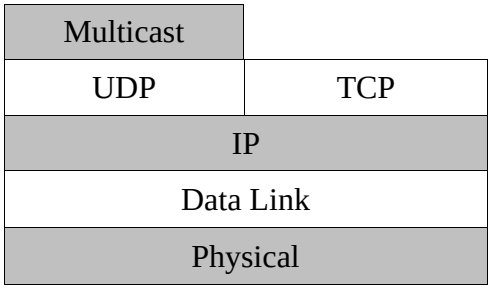


Figure 2 - Multicast TCP/IP Stack

2.1.2 Push versus Pull Technology

To receive data over multicast, a user joins a multicast group. However, once in the group, there is no user initiated interaction. That is, no request needs to be made to the sender of

the group to begin receiving data making multicast a 'push' technology. Probably the oldest and most widely used push technology is email. This is a push technology because you receive mail whether you ask for it or not -- that is, the sender *pushes* the message to the receiver. Contrary to push technology, with pull based technology, a user must make a request prior to being served. The World Wide Web is based on pull technologies where a page is not delivered until a browser requests it.

Push technology is very desirable because it is much more scalable than traditional pull technologies. Consider, for example, a news ticker application which sits on your taskbar and pulls updates every few seconds or minutes. This kind of traffic would quickly clog networks and render them useless. PointCast is such a technology and offers news headlines when the screen saver comes on or when the user calls up this feature. The news updates that are "pulled" automatically in the background to ensure that no stale news is provided. Many large organizations have found that PointCast wastes network resources and have ordered employees to remove it. PointCast would benefit greatly from a Multicast based push technology.

2.1.3 Multicast IP Addressing

Multicast IP uses different IP addresses than are used for point-to-point communications. Class A,B and C addresses have a host and network component. In contrast, Class D IP addresses used for multicast transmission contain only the destination multicast group address. Furthermore, Class D addresses are used on a session by session basis whereas Class A,B and C addresses remain static for the specific computer[5].

The multicast address space occupies the range from 224.0.0.0 to 239.255.255.255. The IANA[6] maintains lists of registered users and assigns new numbers for new uses. The range from 224.0.0.0 to 224.0.0.255 is reserved for permanent assignment for various applications, including routing protocols.

Some well known address have been assigned to groups as follows:

- All systems on this subnet: 224.0.0.1

5 This is not entirely true. DHCP can be used to dynamically allocate address on a host by host basis.
6 International Assignment Number Authority assigns new IP numbers for new uses. Complete list of may be found here: http://www.iana.org/assignments/multicast-addresses

- All routers on this subnet: 224.0.0.2
- All DVMRP routers: 224.0.0.4
- All OSPF routers: 224.0.0.5
- All OSPF designed routers: 224.0.0.6
- All RIP2 routers: 224.0.0.9
- All PIM routers: 224.0.0.13
- All CBT routers: 224.0.0.15

The remaining multicast addresses, 224.0.1.0 to 239.255.255.255, are either assigned to various multicast applications or currently unassigned. A subset of this set, from 239.0.0.0 to 239.255.255.255[M 1998] is reserved for various “administratively scoped” applications but can nevertheless be used by applications.

2.1.4 Mapping Multicast Address to MAC Addresses

Network interface cards have hardware level multicast support. The IANA has been allocated a reserved portion of the IEEE-802 MAC-layer multicast address space. All reserved multicast addresses begin with 01-00-5E (hex). In order to take advantage of this a simple conversion has been established to convert multicast IP addresses to hardware MAC addresses.

Mapping a class D address to a Ethernet multicast address is obtained by placing the low-order 23 bits of the class D address into the low-order 23bits of IANA's reserved block. With this conversion it is possible that two multicast addresses overlap. In fact, for every different multicast address, a collision of 32 different addresses is possible.

In practice, this issue is not a problem because the chance of an overlap is very small. However, even if an overlap occurs, different applications should have different UDP port numbers so packets will be properly directed to the correct application even if they share the same multicast MAC address.

2.1.5 Scoping Multicast Addresses

Time-to-live (TTL) is a value in an Internet Protocol (IP) packet that tells a network router whether or not the packet has been in the network too long and should be discarded. For a number of reasons, packets may not get delivered to their destination in a reasonable length of time. For example, a combination of incorrect routing tables could cause a packet to loop endlessly. A solution is to discard the packet after a certain time and send a message to the originator, who can decide whether to resend the packet. The original idea of TTL was that it would specify a certain time span in seconds that, when exhausted, would cause the packet to be discarded.

Most current IP multicast implementations achieve some level of scoping by using the TTL field. With the Mbone, the TTL field is typically used to confine Multicast traffic to some administratively defined topological region. The scoping field is used to control the distribution of multicast traffic with the objective of easing stress on scarce resources, or to achieve some kind of improved privacy or scaling properties. Administratively scoped addresses are in the range from 239.0.0.0. to 239.255.255.255[M 1998]. These addresses provide two basic functions:

1. Packets addressed to an administratively scoped multicast address do not cross configured administrative boundaries.
2. Administratively scoped IP addresses are assigned locally and thus need not be unique across administrative boundaries, permitting their reuse.

A multicast router will only forward a multicast datagram across an interface if the TTL field in the IP header is greater than the TTL threshold assigned to the interface. Table 2-1 lists the conventional TTL values that are used to restrict the scope of an IP multicast packet. For example, a multicast datagram with a TTL of less than 32 is restricted to the same site and should not be forwarded across an interface to other sites in the same region.

Initial TTL	***Scope***
0	Restricted to the same host
1	Restricted to the same subnetwork
32	Restricted to the same site
64	Restricted to the same region
128	Restricted to the same continent

Initial TTL	*Scope*
225	Unrestricted in scope.

Table 1 - Scoped Multicast

In this manner administrators can configure their independent local area networks into an autonomous system. A corporation's complex internal network might be a single AS, as may the network of a single Internet service provider. Table 1 shows a simple network with two autonomous systems.

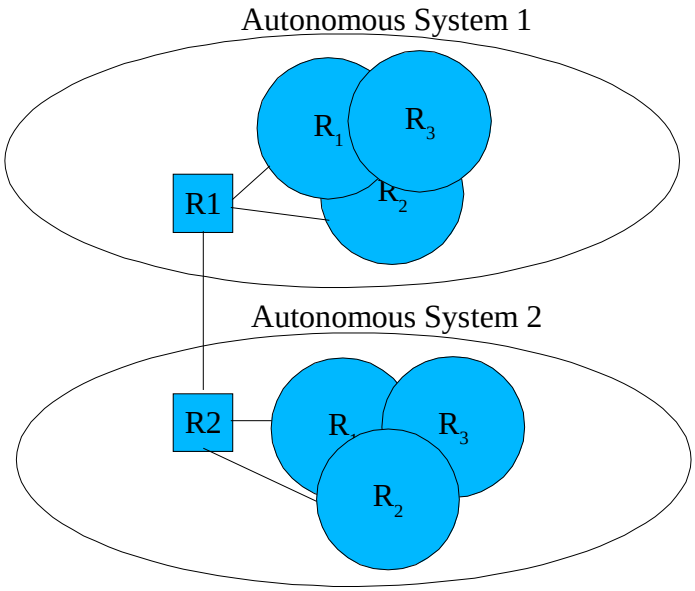


Figure 3 - Autonomous Systems

2.1.6 Multicast Transmission

Multicast transmission can be classed into two sections:

- The sender is on the same local are network as the receiver.
- The sender is on a network which is separated by one or more routers.

In the first case, the transmission and reception of multicast frames is a relatively simple process. The source station simply addresses the IP packet to the multicast group while the network interface card maps the Class D address to the corresponding IEEE-802 multicast address, and the frame is sent. Receivers that wish to receive the frames notify their IP layer that they wish to receive datagrams addressed to the group. Scoping the datagrams with a TTL of 1 would allow only this type of multicast transmission.

When sender and receiver are separated by one or more routers things become more complicated. In order for the sender to send packets, the routers connecting sender and receiver are required to implement one of many multicast routing protocols. These protocols permit the construction of multicast delivery trees and support multicast datagram forwarding. Also, in order for the receiver to receive multicast traffic, the routers connecting sender and receiver must implement a group membership protocol. This protocol allows routers to learn about the existence of group members on its directly attached subnetworks.

2.1.7 Multicast Routing Protocols

Routers in an inter-network use multicast routing protocols to optimally route multicast packets between inter-connecting networks. This approach is similar to unicast best-effort routing protocols that serve as the backbone for all connections in TCP/IP networks.

The basic premise behind routing is to find the lowest-cost path between any two network nodes. There are two basic algorithms used today to accomplish routing in an efficient manner: *distance vector* and *link state[PD 2000]*.

2.1.7.1 Distance Vector

Distance vector routing depends on each node's ability to detect all other nodes. It maintains this information, called reachability information, in a vector. Hence the name Distance Vector Routing. Once a node builds a distance vector of reachability information, it distributes this information to all other adjacent nodes. These nodes then update their reachability information and redistribute. Reachability information consists of a table of { Destination, Cost, NextHop } tuples. In this manner a node using DVRP can extrapolate the shortest path to any other node.

Unfortunately in certain very unlikely circumstances DVRP can prevent the network from stabilizing. This is called the count to infinity problem and only occurs if a certain link goes down during the stabilizing process. Fortunately, there are solutions to this problem. Namely *split horizon* and *split horizon with poison reverse*.

2.1.7.2 Link State

Link state routing is based on the same premise as DVRP. Every node knows how to reach all its directly connected neighbors and this allows information to be distributed across the network. Additionally, LSP differs from DVRP because each node in LSP knows the entire topology of the network and can construct a spanning tree rooted at that node traversing the entire network. However the difference between DVRP and LSP is the use of reliable flooding. Reliable flooding is the process of making sure that all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes. The drawback of flooding is the use of network bandwidth. When a node detects a link change it announces to all other nodes the state change of the link. This in effect floods networks and consumes bandwidth.

These protocols allow any node on an internetwork to successfully send unicast packets to any other node on the same internetwork. As mentioned earlier, Multicast networking is concerned with the ability of a source to send a single packet to a *multicast address*, and for the network to deliver a copy of that packet to each of the multicast group members. Current routing protocols as defined above cannot meet this requirement and a new set of multicast routing protocols have been defined.

There are a number of different algorithms that may potentially be employed by multicast routing protocols:

- Flooding
- Spanning Trees
- Reverse Path Broadcast (RPB)
- Truncated Reverse Path Broadcasting (TRPB)
- Reverse Path Multicasting (RPM)
- Core Based Trees

Multicast routing protocols can be broken down into two categories:

- Dense mode
- Sparse mode

Dense mode multicast routing protocols require some form of flooding of datagrams to the network to find multicast routes. This tactic is most suitable for areas with a dense concentration of group members. In contrast, *sparse mode* multicast routing protocols are best suited for widely dispersed group members.

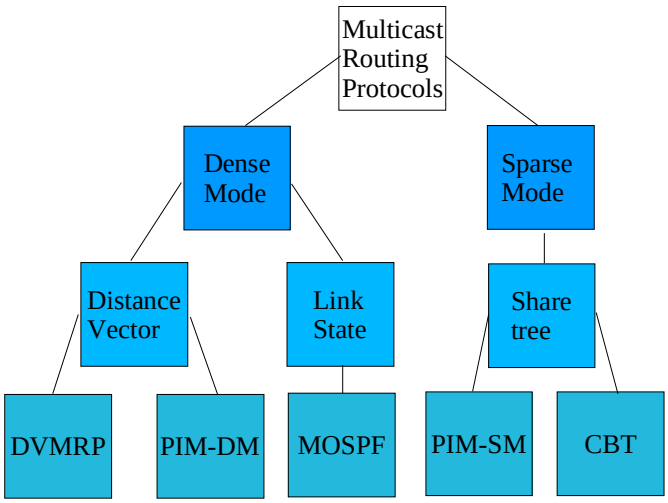


Figure 4 - Multicast Routing Protocols

2.1.7.3 Link State Multicast

As stated above, with link state routing, routers monitor the state of their immediately connected neighbors and send update messages to all other routers whenever the state changes. This way each router has sufficient information to reconstruct the topology of the network. Once the entire topology of the network is described, a router can use Dijkstra's algorithm to compute the shortest path spanning tree rooted at itself.

In order for link state routing to support multicast each node must additionally store group membership information. Group membership information is then added to the link state information and stored on each node as part of the spanning tree. However, each node must somehow discover and persistently maintain a list of active group members. This is done through beacon announcement packets. Each host periodically announces to its router its intent to remain or become part of a group. The absence of such messages would indicate to a router that no more members exist, and the router would prune that group from the

multicast spanning tree.

Multicast link state is supported by a number of major router vendors, particularly those that have embraced link-state routing as a unicast routing protocol.

2.1.7.4 Distance Vector Multicast

Distance vector multicast routing is the oldest routing protocol. It was first defined in RFC 1075 and then updated in a recent Internet Draft[T 1998]. It is the routing protocol first used to implement the Mbone, and remains the dominant multicast routing protocol used there.

Routers using distance vector multicast algorithm use RPB[7], which adds a small amount of overhead to the basic distance-vector algorithm, until a particular multicast address becomes active. RPB basically forwards a packet on all of its links if and only if that packet came from its shortest path. This strategy effectively floods the network outward from the root and does not loop back. Furthermore, a packet is removed from a LAN by a router if it is a duplicate. This only occurs if two or more routers are connected to the same LAN.

When a group becomes active, routers that are not interested in receiving packets addressed to that group speak up using RPM[8], and that information is propagated to the other routers. A router identifies that it has no members of a group and generates a “no members here” message which it propagates up the shortest-path tree.

The protocols reviewed thus far are labeled dense mode protocols because they rely on the routers announcing their negative acknowledgments about group membership. This flooding uses up network bandwidth.

The rapid growth of the Mbone is beginning to place increasing demands on its routers. As the number of subnetworks continues to increase, the size of the routing tables and of the number of periodic update messages will continue to grow. At this rate, the processing and memory capabilities of the Mbone routers will eventually be depleted and routing on the Mbone will fail.

7 RPB stands for Reverse Path Broadcast
8 RPM Stands for Reverse Path Multicast

2.1.7.5 Protocol Independent Multicast (PIM)

PIM was developed in response to the scaling problems of existing multicast routing protocols. In particular it was recognized that the existing protocols did not scale well in environments where a relatively small proportion of the routers want to receive traffic for a certain group. In essence, broadcasting traffic to all routers until they explicitly ask to be removed from the distribution is not a good design choice if most routers don't want the traffic in the first place. PIM approaches this problem from two directions:

- Protocol Independent Multicast Sparse Mode (PIM-SM)
- Protocol Independent Multicast Dense Mode (PIM-DM)

PIM-SM

In PIM Sparse Mode, routers explicitly join and leave the multicast group using PIM protocol messages known as Join and Prune. In general, PIM uses a number of routers in a domain as candidate *Rendez-vous Points* and defines a set of procedures by which all the routers in a domain can agree on the router to use as the RP for a given group. Of course these procedures must deal with complex distributed systems occurrences such as network partitions and RP router failure.

As a result of PIM Join messages, two types of trees can be built:

1. Shared Trees
2. Source Specific Trees

Shared trees use the (*,G) tuple forwarding state also known as the any source multicast. This forwarding state identifies the source as a member of group 'G' and the source being a wildcard '*'. Therefore, the root of the tree is the PIM Sparse Mode RP. In the case of source-specific trees, the forwarding state is the (S, G) tuple. With this tuple the root of the tree is the PIM Designated Router for the source S sending to group G.

All the mechanisms for building and maintaining trees depend on whatever unicast routing protocol is used in that domain. The formation of trees is entirely determined by the paths

that Join messages follow, which is determined by the choice of shortest paths made by unicast routing. Thus, PIM is “unicast routing protocol independent.”

2.1.8 Joining Multicast Groups

Along with the different multicast routing protocols, multicast communication requires a group membership protocol. Hosts join multicast groups using a protocol called Internet Group Membership Protocol (IGMP). As shown in figure 4, hosts use this protocol to notify a router on their local network of their desire to receive packets sent to a certain multicast group. This protocol is also used by the local area network router to query all hosts and determine the current members of a group.

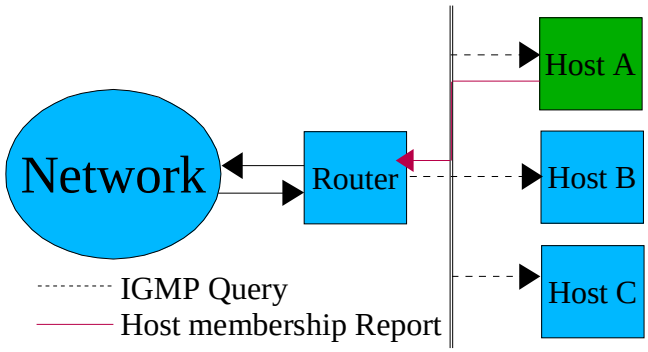


Figure 5 - Multicast Group Membership

2.1.8.1 Internet Group Management Protocol (IGMP)

IGMP is vaguely similar to the Internet Control Management Protocol (ICMP). It comes in three flavors:

- IGMPv1
- IGMPv2
- IGMPv3.

IGMPv1, as specified in RFC 1112[D 1989], is the most commonly used membership protocol today. The mechanism of the protocol allows a host to inform its local router that it wishes to receive transmissions addressed to a specific multicast group. Also, routers periodically query the LAN to determine if known group members are still active. A router does not need to know the exact host in the group in that sub-network; it needs to know only that at least one host belongs to that group on the specified network. This feature reduces the traffic required on that subnetwork.

The total latency in setting up a group depends on the sum of the time to notify the nearest router of the joining member, plus the time to set up the multicast routing by the routers in the network. However, the latency to leave a group is longer, as the timeout needs to expire based on the absence of a response to an IGMP Query before leaving member is recognized. When a router identifies the absence of members in a group, it tears down that multicast tree. This takes approximately one minute for a typical configuration and is the main difference between the different versions of IGMP.

IGMPv2

IGMPv2 defines a new type of query message – the group specific query message. It allows a router to transmit a query message to a specific multicast group on a particular subnet rather than to all multicast groups. Also IGMPv2 adds the ability to create an explicit *leave group* message that can greatly reduce the latency for leaving a group in IGMPv1.

IGMPv3

The primary feature added in IGMPv3 is the capability for hosts to select only specific sources of multicast traffic to receive. This feature gives receivers more control over which sources they allow to deliver information to them. Additionally the leave group message from IGMPv2 has been improved by allowing a host to specify the IP address of any source group pair it wishes to leave.

2.1.8.2 Multicast Deployment

The benefits of multicast are clear on many levels. However, it has not been deployed to a greater extent in major networks. One of the major reasons for the reluctance to adopt multicast is simply resistance to change. Network operators are already overwhelmed with the task of keeping their own private networks operational. It would not be a trivial task to send these personnel into the field to turn on multicast in routers everywhere and ensure their functionality.

One important issue is general lack of test tools for troubleshooting network problems with multicast. A multicast distribution tree as created by one of the existing multicast routing protocols, can be very complicated. If any node experiences a problem, all downstream nodes are affected.

Another major issue associated with multicast transmission, specifically multimedia streaming, is quality of service (QoS). To gain high-quality multimedia viewing, high-bandwidth streams of more than 1Mbps are needed and QoS has to be provided for each stream.

The IPMI commissioned a report in 1997 [IPMI] that listed the “top ten” multicast barriers:

1. Limited application software integrating IP multicast
2. Overall network effects of new content not understood by network planners.
3. Corporate network architects and network service planners who perceive that their networks are operating at capacity and that adding new multimedia content will push existing networks and applications beyond that capacity.
4. Remote sites connected at 56Kbps, which will limit the functionality of applications

targeting users at those sites.

5. Security concerns
6. Limited Administrative tools for IP multicast networks.
7. Conferencing, collaboration, and distance learning applications, which may create a new paradigm that end users resist.
8. Limited ISP implementation that restrict consumer and business application expansion potential.
9. Multiple multimedia content distribution standards coupled with interoperability concerns among IP multicast products.
10. "Push" technology vendors have created confusion.

2.2 Multimedia Streaming

A key characteristic of time-based media is that it requires timely delivery and processing. Once the flow of media data begins, there are strict timing deadlines that must be met, both in terms of receiving and presenting the data. For this reason, time-based media is often referred to as streaming media – it is delivered in a steady stream that must be received and processed within a particular timeframe to produce acceptable results. For example, when a audio clip is played, if the data cannot be delivered quickly enough, there might be off pauses and delays in playback. On the other hand, if the data cannot be received and processed quickly enough, movies might appear jumpy as data is lost or frames are intentionally dropped in an attempt to maintain the proper playback rate.

A media stream is the media data obtained from a local file, acquired over the network, or captured from a camera or microphone. Media streams often contain multiple channels of data called tracks.

A media stream can be identified by its location and the protocol used to access it. For example, a URL might be used to describe the location of an MP3 file on a local computer. On the other hand, if the same file is on a web server, the file can be accessed through the HTTP protocol.

The quality of the stream of media depends on several factors, including:

- The compression scheme used
- The processing capability of the playback system
- The bandwidth available

Traditionally, the higher the quality, the larger the file size and the greater the processing power and bandwidth required. Bandwidth is usually represented as the number of bits that are transmitted in a certain period of time – the bit rate.

2.3 Multimedia Streaming Protocols

Multimedia streaming protocols are often associated with multicast networks. High-quality multimedia streams that include video have an incentive to use multicast if it is delivered to many recipients simultaneously because of the high bandwidths involved. The multimedia real-time streaming applications that fit best with today's Internet infrastructure and have the greatest value are the ones that are receiving the most attention today. For example, Internet telephony is viewed as a huge new use of the Internet. Similarly, audio on a Web site can provide added appeal. Although Web site multimedia can make the sites more interesting, the bandwidth constraints of today's Internet have limited its acceptance at this time.

In the one-to-many category of multicast applications, Internet radio channels are available over the Mbone and hint at the possibilities if the Internet became fully multicast-enabled. Today's most important multicast multimedia streaming applications are driving the development of the technology that will bring these applications into the mainstream.

2.3.0.1 Multimedia Streaming Formats.

Audio is generated from analog signals by some form of sampling technique. A microphone converts the sound energy into an analog electrical signal, and a speaker converts this electrical signal back into analog form. The electrical signal is converted to a digital signal as it passes through an analog to digital converter (ADC). Conversely, the digital signal is converted to analog as it passes through a digital to analog converter (DAC). As an example the telephone network characterizes voice with a 300Hz to 3.3KHz

bandwidth sampled at 8Kz with 7bits per sample. This set-up yields the standard telephone voice channel data rate of 56Kbps.

This data once sampled is usually compressed before transmission through the network. Compression/Decompression is accomplished through the use of *codec*s. A huge number of codecs are available today all using different algorithms each with it's own trade off of speed versus compression. Each codec has certain input formats that it can handle and certain output formats that it can generate. In some situations, a series of codecs might be used to convert from one format to another.

There are two broad classes of compression algorithms: lossless and lossy. Lossless compression algorithms produce compressed data that can be decoded to output that is identical to the original. Zip is a very common example of a lossless compression format. FLAC is a lossless compression format that is specifically designed for audio.

The other type of compression algorithm is called lossy. This form of compression is very popular with multimedia data such as pictures, movies, and sound. Since these types of information are perceived by humans with imperfect senses, the original data does not have to be reproduced exactly. Some of the information in the original file can actually be discarded because we wouldn't notice it even if it was there. Lossy codecs can achieve much higher compression than lossless codecs by intelligently discarding unneeded information. In most cases, some loss of quality can be tolerated so even more data can be discarded, further increasing compression. MP3, RealAudio, and Vorbis all use lossy audio compression.

MPEG Audio Layer-3

MPEG Audio Layer-3, usually called MP3 was first derived by Fraunhofer IIS-A in 1987. MP3 describes the compression of audio signals using high performance perceptual coding schemes addressing the perception of sound waves by the human ear. It specifies a family of three audio coding schemes, simply called:

- Layer-1
- Layer-2
- Layer-3

From Layer-1 to Layer-3 the complexity increases, the overall codec delay increases and performance increases.

By using MP3 coding, you may shrink down the original sound data from a CD by a factor of 12, without losing sound quality. Factors of 24 and even more still maintain a sound quality that is significantly better than what you get by just reducing the sampling rate and the resolution of your samples. While still maintaining the original CD sound quality one may achieve a typical data reduction of:

1:4	by Layer 1 (corresponds with 384 kbps for a stereo signal)
1:6...1:8	by Layer 2 (corresponds with 256..192 kbps for a stereo signal)
1:10...1:12	by Layer 3 (corresponds with 128..112 kbps for a stereo signal)

By exploiting stereo effects and by limiting the audio bandwidth, the coding schemes may achieve an acceptable sound quality at even lower bitrates. MP3 is the most powerful member of the MPEG audio coding family. For a given sound quality level, it requires the lowest bitrate - or for a given bitrate, it achieves the highest sound quality.

Sound Quality	***mode***	***bitrate***	***Reduction ratio***
Telephone sound	mono	8Kbps	96:1
FM radio	stereo	56...64 Kbps	26...24:1
CD	stereo	112...128Kbps	14...12:1

[MP3]

To further enhance the ease of use of MP3 files, an ID3 tag has been defined. Although no official body has take responsibility, the ID3 standard it is widely used and most audio players support it. The ID3 is an information field embedded into MP3 files used to save information such as the Title, Artist, Album etc. The MP3 standard does not provide a means of storing such information itself so ID3 tags are used to accomplish this.

The ID3 tag fields are in the last 128 bytes of the file and their basic structure is as follows:

Field Name	*Length (bytes)*	*Description*
Tag	3	This field simply contains the string "TAG" This is used to signal the beggining of an ID3 tag.
Song Name	30	This field stores the name of the Song.
Artist	30	This field contains the name of the artist.
Album	30	This field is for the name of the album.
Year	4	This field stores the year of release of the song.
Comment	30	This field stores any comments.
Genre	1	This is an unsigned byte field storing the offset of genres predefined in a descriptor list. The list shown below.

Table 2 - ID3 Tag Contents

Ogg Vorbis

Another audio format is Ogg Vorbis. Ogg Vorbis is is a new audio compression format. It is roughly comparable to other formats used to store and play digital music such as MP3, VQF, AAC, and other digital audio formats. It is different from these other formats because it is completely free, open, and unpatented. Vorbis is the name for the specific audio compression scheme used to create Ogg Vorbis files. It is part of the Ogg project, which is a blanket project designed to create a fully open multimedia system. It has been designed to completely replace all proprietary, patented audio formats. This means that you can encode all your music or audio content in Vorbis and never look back.

Since MP3 is a “lossy” format much of the sound data is removed when MP3 files are created. This results in a file with inferior sound quality to a CD. Vorbis is also a "lossy" format, but uses superior acoustic models to reduce the damage. Thus, music released in Vorbis will sound better than a comparably sized MP3 file. Furthermore streaming using MP3 entails certain payment issue because MP3 is a patented technology. Vorbis is patent and license-free, so there is no payment for anyone in order to sell, give away, or stream Ogg Vorbis music.

2.3.0.2 Real-Time Streaming Protocols

As mentioned earlier, multicast communication is done over the universal datagram protocol (UDP). UDP however, does not make any guarantees for error free delivery or packet ordering. Furthermore, UDP does not provide any congestion control mechanism. Unlike UDP, TCP has congestion avoidance features. These features, known as end-to-end congestion control, allow TCP to react to observable events that occur on a network[PD 2000]. In order to stream multimedia reliably and efficiently over a network connection, a streaming protocol should make certain guarantees:

1. The ability to communicate the choice of coding scheme used in the stream. This would allow clients to identify the type of stream.
2. A timing relationship between packets for appropriate playback of the stream content.
3. Identification of packet loss. This would allow for certain congestion control measures.
4. Frame boundary indication.
5. Efficient use of bandwidth.
6.

Real-Time Transport Protocol (RTP)

RTP was developed in the IETF and is in widespread use. The RTP standard [RTP] actually defines a pair of protocols, RTP and the *Real-Time Control Protocol* (RTCP). RTP is used for the exchange of multimedia data while RTCP is used for the exchange of control information of that data.

RTP uses an even port number while RTCP control information uses the next higher port number.

For each type of application RTP defines a profile and one or more formats.

- *The Profile*: provides a range of information that ensures a common understanding of the fields in the RTP header for that application.
- *The format*: explains how the data that follows the RTP header is to be interpreted.

Since RTP's development, a number of profiles have been defined. For example, JPEG video profile for RTP is defined in RFC 2035.

RTP Header Format

2bits	1bit	1bit	4bits	1bit	7bits	16bits
Version	Padding	Extension	Contributing Sources	Marker Bit	Payload Type	Sequence Number
Timestamp						
Synchronization Source identifier						
Contributin Source identifiers...						
Extension Header						
RTP playload						

Figure 6 - RTP Packet ***[RTP]***

Figure 6 shows the RTP header format**[RTP]**. The fields to note are the *payload type, timestamp* and *sequence number*.

Payload type:

The payload type indicates what type of multimedia data is carried within the payload of the packet. The exact usage of the payload field is defined in the profile.**[RTP]**

Timestamp:

The timestamp field is there to enable the receiver to play back samples at the appropriate intervals and to enable different media streams to be synchronized. Because different applications may require different granularities of timing, RTP itself does not specify the units in which time is measured. Instead, timestamp is just a count of evenly spaced ticks.**[RTP]**

Sequence Number:

This field is used to enable the receiver of an RTP stream to detect missing and misordered packets. RTP however takes no action when a missing packet is detected. Rather it is left to the application to decide what to do when a packet is lost because the decision is likely very application specific.**[RTP]**

Real Time Control Packets (RTCP)

Each RTP flow is supplemented by RTCP packets. A number of different RTCP packet types exist. RTCP packets provide the relationship between the real-time clock at a sender and the RTP media timestamps. They also provide textual information to identify a sender in a conference from the source ID.

In IP multicast, sources may send to a multicast group without being a receiver in that group. For many applications however, it is useful to know who is listening to the transmission and whether the media flows are reaching receivers properly. RTCP provides approximate group membership information through periodic multicasts of session messages that give, in addition to information about the recipient, information about the reception quality at that receiver. This information may be useful for rate-adaptive applications, which may use performance data to decide to use a more aggressive compression scheme to reduce congestion, or to send a higher-quality stream when there is little congestion. It can also be useful in diagnosing network problems. Fortunately, these messages are restricted in rate so that, as the membership of a particular multicast group grows, the rate of RTCP messages remains constant. The end result of session messages is for any sender to have a general idea of the current listeners in the multicast group.

RTP Translators/Mixers

RTP also includes a provision to support "translators" and "mixers." These devices interconnect different networks that may support different transports or codecs. A translator passes along data streams from different sources as separate entities whereas a mixer combines them to form one new stream.

The design of RTP embodies an architectural principle known as *Application Level Framing* (ALF). This principal was put forth by Clark and Tennenhouse in 1990[ALF] as a new way to design protocols for emerging multimedia applications. It is because of ALF considerations that RTP leaves so many of the protocol details to the profile and format documents that are specific to an application.

Although RTP contains a considerable amount of functionality that is specific to multimedia applications, it is still viewed as a transport protocol. Therefore the TCP/IP stack illustrated in figure 1-1 has now gained a new layer that sits on top of the multicast

layer.

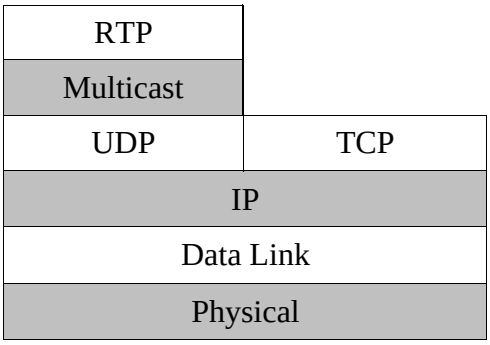


Figure 7 - TCP/IP Multicast Stack

Real-Time Streaming Protocol (RTSP)

The Real-Time streaming Protocol is a control protocol very similar to HTTP/1.1. It is used to establish and control one or more time-synchronized streams of continuous media, such as audio and video. In general is does not deliver the continuous streams itself, although interleaving of the continuos media stream with the control stream is possible. Essentially, RTSP acts as a "network remote control" for multimedia servers, similar to a VCR control for television.

RTSP sessions are not tied to a transport connection, such as a TCP connection. During an RTSP session, a client may open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it may use a connectionless transport protocol such as UDP.

The streams controlled by RTSP may use RTP, but the operation of RTSP does not depend on the transport mechanism used to carry continuos media.

RTSP is intentionally similar in syntax and operation to HTTP/1.1, allowing most extension mechanisms to HTTP to be added to RTSP. The protocol support the following operations:

- Retrieval of Media from a Media Server:

  The client can request a presentation description via HTTP or some other method. If the presentation is being multicast, then the presentation description contains the multicast addresses and ports to be used for the stream. If the presentation is to be

sent only to the client via unicast, then the client provides the destination for security reasons.

- Invitation of a Media Server to a conference

  A media server can be “invited” to join an existing conference, either to play back media into the presentation or to record all or a subset of the media in a presentation.

- Addition of Media to and Exiting Presentation

  Particularly for live presentation, it is useful if the server can tell the client when additional media becomes available.

2.4 Recommender Systems

The networked world contains a vast amount of data. Visitors face the impossible task of retrieving information that matches their preferences, information that they would find interesting or information that they would find relevant. This is indeed becoming more and more of a problem as the web grows and as new content continues to spring up. In order to help users sift through information, they rely on Recommender Systems. Recommender systems are learning systems that make use of data representing multi-user preferences to try to predict the preferences of new items. The key problem of information overload facing computer scientists is how to create recommendations based on this vast amount of information available. This is a problem because the vast majority of data available is geared for human, and not machine, consumption.

In essence, Recommender Systems try and group similar users into sets intelligently.
Based on the information in **Figure 8**, a recommender system would recommend assets 9

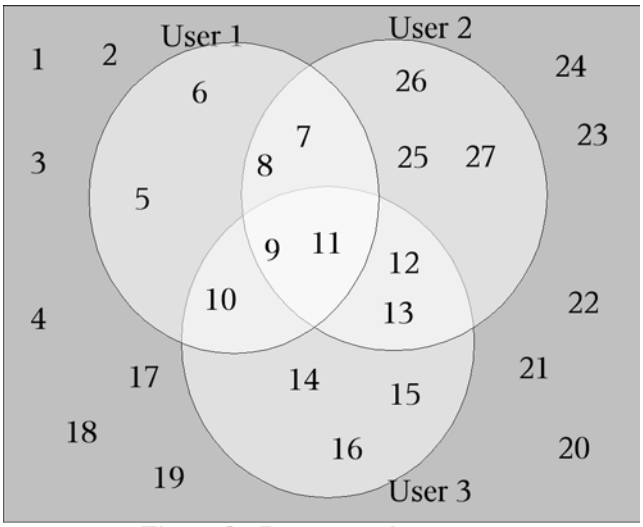


Figure 8 - Recommender system

and 11 to all three users, assets 8 and 7 to users 1 and 2, assets 12 and 16 to users 2 and 3 and so on.

Various Recommender Systems have been invented to address the information overload problem. Most use either the *collaborative-filtering* or the *content-based* approach. Content-based filtering analyzes the content of information that has been rated by a user to create a profile of that user's interests. Collaborative approaches find and recommend information sources for an individual user that have been rated highly by other users who have a pattern of ratings similar to that of the user. Another type of recommender system is the hybrid system. This system attempts to combine the advantages of both content based and collaborative based recommendation systems. With enough information, Recommender Systems personalize a site on a per user basis. Personalization is the ability to provide content and services tailored to individuals on the basis of knowledge about their past and current preferences and behavior.

Even with the available techniques, Recommender Systems must face certain problems which are inherent in any recommendation system.

2.4.0.1 The Cold Start Problem

One difficult problem commonly faced with Recommender Systems is the cold-start problem. This problem arises from new Recommender Systems with no data yet available

for the user or the asset. Poor performance resulting from a cold-start can deter user uptake of a recommender system. This effect is self-destructive, since the recommender never achieves good performance because users don't join and participate for long enough. There are two main cold start scenarios:

- *New-system cold-start*: This problem is where there are no initial ratings for assets. In this case there can be no profiles for users and hence no basis on which to recommend assets.
- *New-user cold start*: This problem occurs even after the system has been up and running for a while. A new user joins the system but a profile for this user does not yet exist. In this case most recommenders perform poorly.

Typical Recommender Systems fail in these situations because insufficient data is available to recommend assets to users. In these cases a more creative but less used solution, such as demographic-based recommendation or temporal based recommendations, must be implemented.

Another possible problem facing Recommender Systems is the issue of privacy. As the recommender system evolves over time it develops usage patterns based on decisions made by the user. Essentially the Recommender Systems learns a user's habits and indexes this data for future retrieval. There exists a potential for this gathered data to be used for advertising or other purposes not known by the user. This poses a possible real concern for Recommender Systems as they become more and more complex.

2.4.0.2 Collaborative Based Filtering

Collaborative based filtering combines the processing power and precision of computers with the intelligence of humans. Users indicate their opinions in the form of ratings on various pieces of information, and collaborative filter correlates the ratings with those of other users to determine how to make future predictions. Additionally, collaborative filtering shares information gathered from one user with other users so they can use them in making their own predictions. In the following example, assets are rated by users A, B, and D. However Asset 3 has only been rated by user B and D. Collaborative filtering would allow for recommendation of asset 3 based on similar ratings of other assets for all three users.

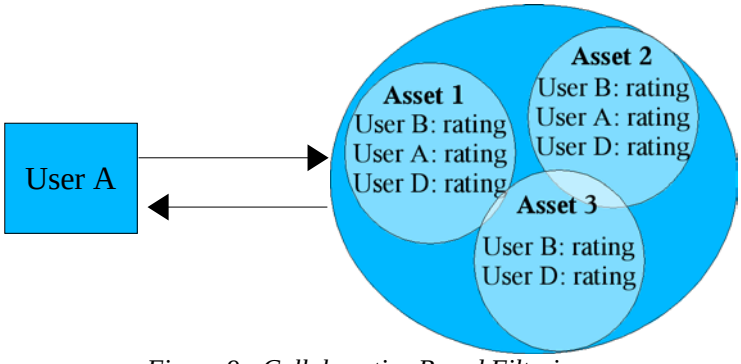


Figure 9 - Collaborative Based Filtering

Most of the successful research and commercial systems in collaborative filtering use a nearest-neighbor model for generating predictions. Automated collaborative filtering systems based on the nearest-neighbor method work in three phases:

1. Users of an automated collaborative filtering system rate assets that they have previously experienced.
2. The system matches the user with other participants of the system who have similar rating patterns. This is usually done through statistical correlation. Neighbors are selected from a list of closest matches.
3. Items that the neighbors have experienced and rated highly, but which the user has not yet experienced, will be recommended to the user. These items will be ranked based on the closeness of the neighbors to the user.

In order to implement collaborative filtering, different algorithms are being used.

Memory based algorithms[BHK 1998] predict the votes of the active user based on some partial information regarding the active user and a set of weights calculated from the user database.

Correlation algorithms [BHK 1998] were used in the GroupLens project where the Pearson correlation coefficient was defined as the basis for the weights.

Vector similarity calculates the similarity between two assets, treating each asset as a vector of properties and computing the cosine of the angle formed by the two frequency vectors.[BHK 1998]

Collaborative filtering has been successfully implemented in many web sites and is being used to recommend everything from books, restaurants and jokes. However, collaborative filtering does suffer from certain drawbacks:

- *Early Rater*: Collaborative filtering cannot provide a good prediction for an asset if it has been recently added to the system because no users have had the opportunity to rate the asset. Moreover, early predictions for the asset will often be inaccurate because there are few ratings on which to base predictions. Similarly an established system may still provide poor predictions for new users as there is no asset data for that user.
- *Sparsity problem*: Often the number of assets will far exceed what any individual can absorb, thus user asset ratings will be very sparse and may not prove to be useful. This will make it hard to find assets that have been rated by other users.
- *Gray Sheep*: In many small to medium sized communities there are individuals that do not benefit from collaborative filtering techniques because their opinions do not match any other users ratings. These users will rarely receive accurate predictions, even after the initial start up phase.

2.4.0.3 Content Based Recommendations

As stated above, content based filtering methods make recommendations by analyzing the description of the items that have been rated by the user and the description of items to be recommended. Content based filters are less affected by the above problems of pure collaborative filters because they use techniques that apply across all documents. Borrowing from the collaborative based recommendations example, content based

recommendations make certain modifications. User rated assets are described by keywords. These keywords are then matches against the whole set of assets to find a possible correlation.

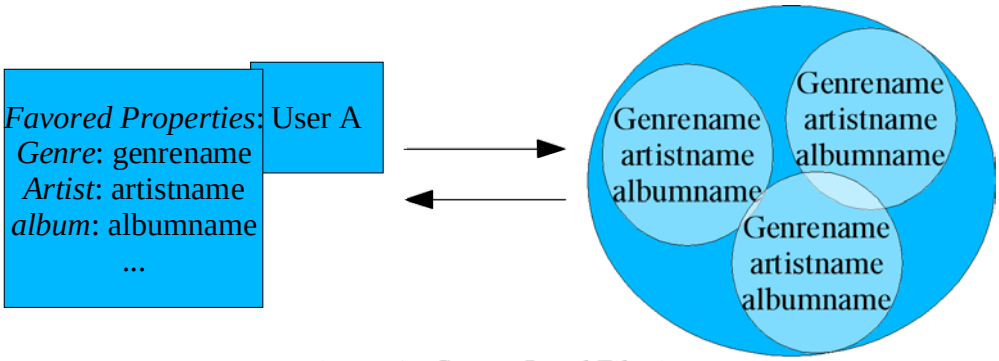


Figure 10 - Content Based Filtering

There are two important sub-problems in designing a content-based filtering system:

1. Finding a correct representation of the asset
2. Create a profile for unseen assets to be recommended.

Once a representation has been found for an asset, a classification algorithm can learn and create a profile to distinguish representations of highly rated assets from other assets. There are a variety of algorithms that have been defined for analyzing the content of an asset and finding regularities in this content that can serve as the basis for making recommendations. Search engines use Rocchio's algorithm[R 1971] to learn the average of the documents that are highly rated. Bayesian classifiers can be used to estimate the probability that a document is liked. The winnow algorithm is designed to identify relevant features when there are many possible attributes.

Despite the advantages of content based filtering it may still prove ineffective in certain situations. Unlike humans, content-based techniques have difficulty in distinguishing between high-quality and low-quality information that is on the same topic. And as the number of items grows, the number of items in the same content-based category increases, further decreasing the effectiveness of content-based approaches. Furthermore, content-based approaches depend on deterministic assets properties. The properties must be clearly defined by humans in order to be indexed by computers. In certain situations it is difficult to determine the correct property which would best describe a give asset.

2.4.0.4 Content based/collaborative based hybrid

Evidence exists which shows the effectiveness of a content-based and collaborative-based recommendation hybrid[BS 1997]. By using a combination of these two techniques one can realize the benefits of content-based filters which including early predictions that cover all items and users, while gaining the benefits of accurate collaborative filtering predictions as the number of users and ratings increases.

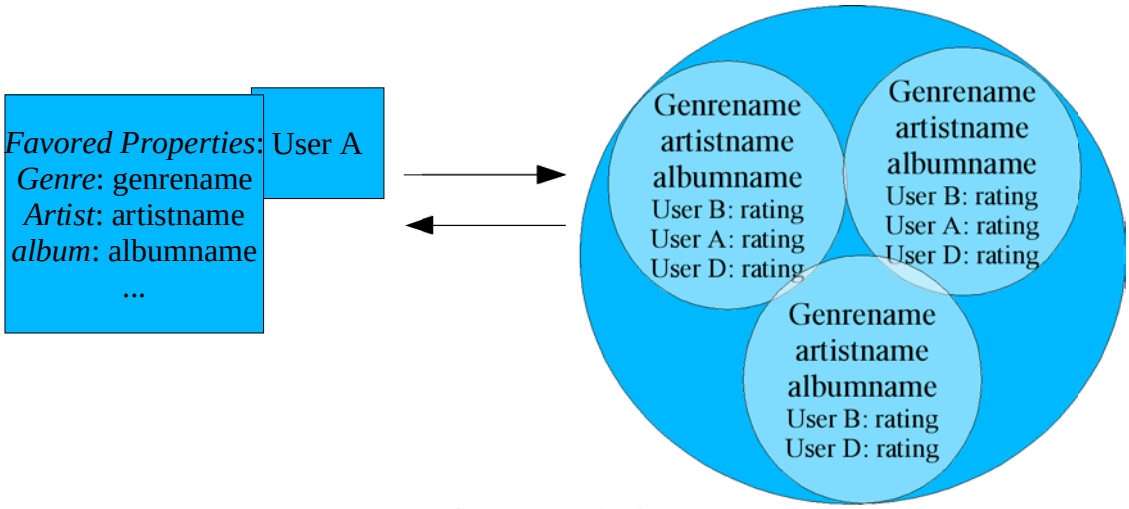


Figure 11 - Content Based Filtering Example

2.5 Web Servers

A web server is a server which dispenses documents that reside on the World Wide Web(WWW). Clients use a program known as a Web browser to communicate with a Web server. On the web server are stored many pages which make up a web document. When a client wishes to view a web page it instructs the browser as to which page is required. The browser will then send a message using a protocol known as HTTP which identifies the page to be accessed and its location. The web server then recovers the page and sends it back to the browser executing on the client. This section will examine how web servers are programmed, how documents are constructed and the different technologies used on the web.

A web server is in effect a very sophisticated file server which dispenses files that contain Web pages, graphics images etc... There are a number of different requests that a user may

issue to the web server. The most common of which is the GET, command which is used to retrieve a page from a web server. An example GET command is shown below:

```
GET /index.html HTTP/1.1
User-Agent: Opera/6.1
Connection: Keep-Alive
Host: localhost
Accept: text/html
```

When the server receives this message it carries out the response that the browser asked for.

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2142
Date: Tue, 10 Sep 2002 13:36:54 GMT
Server: Apache Tomcat/4.0.4 (HTTP/1.1 Connector)
Last-Modified: Thu, 18 Jul 2002 17:40:38 GMT
ETag: "2142-1027014038000"

...<HTML CODE>...
```

When the browser receives the the header lines, and the HTML code it displays the content on the user's screen after interpreting the markup instructions detailed by the HTML tags.

There are two main types of server web pages; *Dynamic* and *Static.* Dynamic pages are first processed and modified before being returned to the user. Static pages are always returned to the user containing the same HTML until modified by hand. As e-commerce applications become more and more common there is an increasing demand for pages that are modified in some way before being downloaded, such as up-to-date stock and share prices. There are a number of technologies such as Perl, PHP, and Java Server Pages, which have emerged for customized web page content dynamically.

2.5.1 Dynamic Web Content languages

There are more than a few programming languages and scripting languages that provide capabilities for dynamic web content. There is Python, PHP, Java, Perl, Tcl just to name a few. Although all server side programming languages have their advantages, Java is used

when true power is required.

2.5.2 Servlets

Java server side programming is done through servlets. Servlets are snippets of code which are loaded into a Web server and which are executed when an HTTP command is processed by the server. Servlets have a number of key features that distinguish them from other server side programming languages:

- They are portable across a wide variety of servers.
- Since servlets are written in Java they can access a huge variety of Java features including CORBA, RMI, Java security facilities and database connectivity facilities.
- In contrast to CGI programming languages like Perl, Servlets are resident in memory. This means that there is significantly reduced overhead on the web server.
- Servlets can maintain state across requests. This means that a servlet can remember data and details from a previous request.
- The object oriented model within Java is very clean compared to other languages like C++ and Perl.

Servlet life cycle

A servlet has a life cycle, and can it find itself in a number of states during the time that it interacts with a Web server.

- The servlet is initialized and is loaded into the Web server's memory.
- The servlet resides in the memory waiting for requests from clients.
- The servlet is destroyed.

2.6 Summary

This chapter reviewed multicast networking and its advantages. Several protocols are

being implemented for TCP/IP multicasting, including upgrades to IP itself for IPv6. Standards for IP multicasting over Ipv4 are provided by several protocols, including the Internet Group Management Protocol(IGMP), Protocol Independent Multicast (PIM), and Distance Vector Multicast Routing (DVMR).

This chapter also reviewed RTP. RTP is designed for end-to-end, real-time delivery of data such as video and voice. RTP, a layer-4 protocol, provides the functionality required for real-time multimedia traffic, including content identification, timing reconstruction, and security. RTCP, a companion to RTP, provides feedback to RTP data sources and recipients.

Also, this chapter looked at Recommender Systems. Recommender systems generally come in two different flavors: Collaborative recommendation and Content based recommendation. A hybrid of the two is also often used. Recommender systems help users cope with the information overload problem that is inherent in any very large set of data such as the internet.

Finally this chapter looked at the web servers. Web servers serve pages to users either dynamically or statically. Many server side dynamic languages exist such as PHP, Perl and recently Java Servlets.

3 Design and Analysis

As stated in the introduction the overall objective of SmartRadio is:

To create a scalable streaming application over the current IP multicast infrastructure, and then, using a Recommender System, suggest and customize, dynamic multimedia content for transmission over this medium.

This objective is too large to address without first braking it down into its constituent components.

1. The users must be able to interact with the recommender system and the streaming multimedia.
2. The web site must follow user click streams, gather user input in the form of asset ratings, and also generate stream suggestions.
3. The streamer must stream multimedia back to the user as directed by the recommender system.

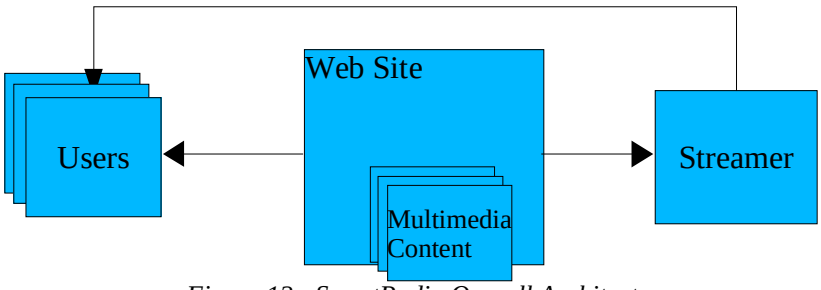


Figure 12 - SmartRadio Overall Architecture

SmartRadio design is consistent with the e-commerce layered web development depicted in Figure 13**.**

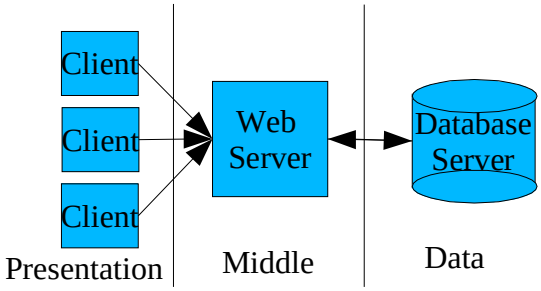


Figure 13 - E-Commerce Three Tier Model

The presentation layer represents the browsers and clients. The middle layer, often referred to as the application layer handles all business logic. The data layer is the persistence layer, it is responsible for maintaining user and stream information in a persistent manner. The SmartRadio application layer uses Java based web server technology to service requests and is the focal point of the recommender system.

This chapter will first analyze each of the constituent parts independently and then look at the system as a whole.

3.1 User Tier

Users interact with SmartRadio in two fundamental ways: They interact with the website, and they listen/receive multimedia streams. These two requirements, although sharing the same goal, are separate technologies and must be analyzed separately.

3.1.1 User and Website interaction

From the user's point of view, the website system should be able to remember preferences and offer the opportunity to rate given assets. Furthermore, this system should provide a convenient form of feedback to the user. Two options exist for meeting this requirement:

1. A SmartRadio plugin for Nullsoft's Winamp.
2. A SmartRadio Web site.

Winamp plugins are Microsoft Windows DLL's that get loaded into Winamp on winamp startup. The benefit of creating a rating system based on this technology is that both the

streaming media and the recommender user interface are the same application. However, Winamp is exclusively a Microsoft Windows product. This would make the SmartRadio user tier operating system centric. Specifically, the SmartRadio user tier would only work on the Microsoft Windows operating systems.

Web browsers are established technologies and are used by everyone connected to the web. According to google.com, there are roughly 2.5 billion web pages online today. Furthermore, the use of the web has been standardized through both the HTTP[9] protocol and the HTML[10] Markup Language. Finally, most popular web browsers available today are free to download and use.

Because the web has already been accepted as an internet medium, SmartRadio should use it as the focal point for interactions between the recommender system and the user.

3.1.2 User/multimedia stream interaction

In order to listen to the multimedia streams being delivered by SmartRadio, the user must use a multimedia-streaming client. This multimedia client must be able to perform the following three tasks:

1. Play media files, specifically MP3 audio files.
2. Support the RTP streaming transport protocol.
3. Be a Fully compatible Multicast client.

The user tier is summarized in Figure 14.

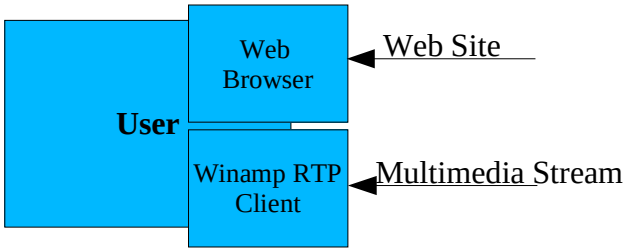


Figure 14 - User Tier Architecture

9 HTTP stands for Hyper Text Transfer Protocol
10 HTML Sands for Hyper Text Markup Language

3.2 Web tier

The web tier must use a powerful, scalable, cross-platform, web server. Furthermore, the web server must have support for a powerful and portable programming language. Java is a fully object oriented, platform independent, easily extensible, highly modular as well as very supported language. These features make Java an obvious choice for large scale applications. A web server supporting the latest version of the Java Servlet Specifications version 2.3 is Tomcat. Tomcat is an open source web server platform.

Running on top of Tomcat, this tier is broken down into three major sections. The first section is the underlying SmartRadio information infrastructure, responsible for gathering user data and presenting data back to the user. The second section is the recommender system itself, responsible for suggestion generation and customized playlist creation. The third section is the stream connector, responsible for correct multimedia timing and playback to the streamer.

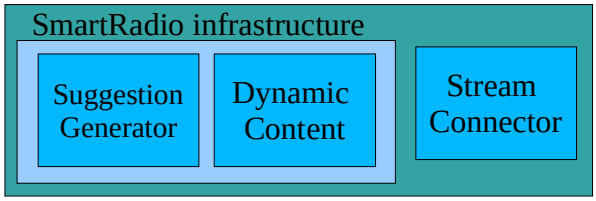


Figure 15 - SmartRadio Web Tier

The SmartRadio web tier as a whole should be designed to provide separation of concerns, decreased code duplication and optimized for centralized control. This in effect makes SmartRadio more easily modifiable. Also the design should help developers with different skill sets to focus their skills and collaborate effectively.

To accomplish these objectives, SmartRadio divides the web tier into three logical sections following the Model View Controller design pattern. Each layer of the MVC design paradigm handles specific tasks and responsibilities.

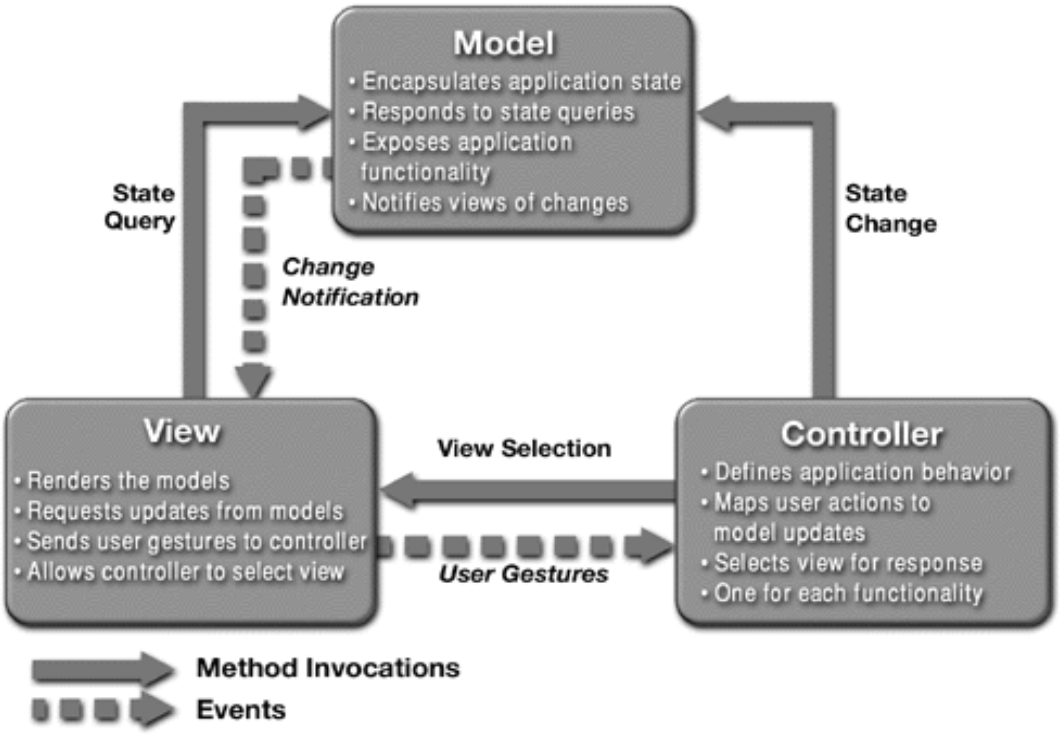


Figure 16 - MVC model[SUN]

A *model* represents business data and business logic or operations that govern access and modification of this business data. Often the model serves as a software approximation to real-world functionality. The model notifies views when it changes and provides the ability for the view to query the model about its state. It also provides the ability for the controller to access application functionality encapsulated by the model.

A *view* renders the contents of a model. It accesses data from the model and specifies how that data should be presented. It updates data presentation when the model changes. A view also forwards user input to a controller.

A *controller* defines application behavior. It dispatches user requests and selects views for presentation. It interprets user inputs and maps them into actions to be performed by the model. In a stand-alone GUI client, user inputs include button clicks and menu selections. In a Web application, they are HTTP GET and POST requests to the Web server. A controller selects the next view to display based on the user interactions and the outcome of the model operations. Some applications use a separate controller for each client type, because view interaction and selection often vary between client types.

The MVC design pattern provides a host of design benefits. MVC separates design concerns that in effect decrease code duplication, centralize control, and make SmartRadio more easily modifiable. This would allow different developers working on different sections of SmartRadio collaborate more effectively. For example, developers could work on improving the user interface while other developers worked on improving the recommender system.

MVC can centralize control of such application facilities as security, logging, and screen flow. Most importantly for SmartRadio is MVC extensibility features. It is easy through the MVC model to add new data sources to an application by creating code that adapts the new data sources to the view API. It is also easy to adapt new client types by adapting the new client type to operate as an MVC view.

If in the future a new SmartRadio recommender client is designed to work through a different presentation layer and completely bypass the web layer, a new view could be implemented with the recommender logic remaining untouched. Moreover, if the SmartRadio model were to be applied to new types of media such as video files or simply new types of audio formats, adding these new data sources would be a simpler task.

In order to help with the MVC design, Struts is used. Struts is a web development framework that builds on Servlets and JSP and makes the development of MVC applications easier. The Struts architecture is broken down into the following areas:

- A controller Servlet that dispatches requests to appropriate Action classes provided by the application developer.
- JSP custom tag libraries and associated support in the controller Servlet that assists developers in the creation of interactive form-based applications.

- Utility classes to support XML parsing, automatic population of Java Beans properties based on Java reflection APIs and internationalization of prompts and messages.

Struts is developed and maintained by Apache, the same people responsible for the tomcat web server. Thus struts and tomcat are extremely interoperable.

Figure **13** illustrates how SmartRadio uses the MVC design paradigm.

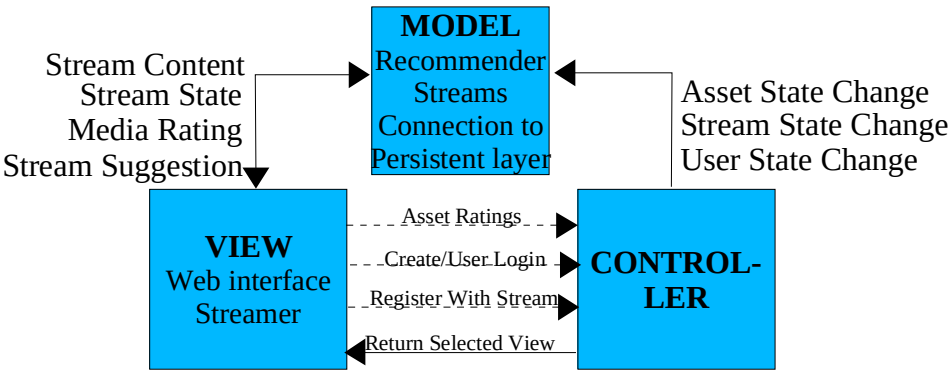


Figure 17 - SmartRadio MVC

3.2.0.1 SmartRadio Model (Mvc)

The SmartRadio Model layer must implement the streams, the recommender system, and the connection to the persistence layer.

- The recommender system itself is responsible for parsing relevant user and stream information and creating user suggestions.
- The streams themselves contain playlists of assets and interface with the recommender system.
- Finally, the Model layer interfaces with the persistence layer.

The connection to the Persistence Layer

The persistence layer is responsible for storing the state of the system. This means that the persistent layer stores stream, user and asset states for later retrieval. It must be developed in such a way that it is easy to later change the persistence layer from one technology to another. Essentially this layer must store:

- All registered members of SmartRadio
- All assets that are part of the system
- Messages sent from the system to users.
- Asset ratings for each user that has rated in the past
- The Stream history.
- The Asset History.
- Stream Ratings.

The Streams

Streams are at first statically created by pulling assets from the database and grouping them into playlists. All streams must be accessible by any user who logs in. When a user selects a stream it is started and the streaming process begins. However, when there are no users currently registered as listening to a stream then that stream must be stopped. When a new user joins a stream, the stream must pick up from where the previous user left off. Therefore, all streams must support start and stop functionality. In order to reach the user, streams are passed through the stream connector, the stream connector in turn sends the stream to the streamer. This will be discussed in the next section.

Recommender System

In order for the recommender system to effectively deal with the cold start problem it uses a hybrid of the collaborative and content based filtering approaches, and a new temporal based filtering approach.

The content based approaches are based on the genre types of each respective file, thus SmartRadio classifies each asset based on its respective genre and uses this criteria to group similar assets into playlists. As users listen to specific streams, their stream usage is tracked and possible recommendations are made. As with any content-based approach the genre type of each asset is pre-defined by a human being. This can be a drawback as not all humans agree on genre types. This is a classic content based filtering problem.

The collaborative-based approach uses user clustering to derive recommendations. Recommendations are based on the similarities between users. This means that songs are migrated from stream to stream to accommodate the current listeners. This in effect addresses the issue raised with content based filtering by slowly perfecting the content of each stream to best suit its listeners.

However, both of these approaches will fail when the system is in the cold start state or when there is a new user. To address this problem, SmartRadio uses the temporal based filtering approach. This approach is based on the correlation between the current time and the most popular streams. After only a few days of operation, SmartRadio can determine the most popular streams for specific time frames and begin recommending on this basis.

On the whole, the SmartRadio recommender system has two responsibilities:

1. To recommend Streams.
2. To build customized playlists.

However, before performing any of the previous steps, the recommender must retrieve asset ratings. This retrieval is done by querying the persistence layer for all asset ratings submitted by users. If an asset has not been rated by a user then the recommender system must infer the ratings. To infer the asset's rating, SmartRadio uses the expected value formula to cluster users:

$$U_x = \bar{U} + \frac{\sum_{J \in \text{Raters of } x} (J_x - \bar{J}) \cdot r_{U,}}{\sum_{J \in \text{Raters of } x} |r_{UJ}|}$$

Figure 18 - User Clustering Formula

This inference can either be done at run time called the lazy approach, or it can be done offline, for example once a week, called the eager approach.

Rating

Before assets can be suggested the user must rate them. The asset rating system designed in SmartRadio allows for two types or rating: asset rating and stream rating.

Assets can either be rated explicitly by the user or can be rated based on whether or not a user has listened to it. There are 5 choices for explicit rating:

1. Hate it
2. Bag song
3. Don't Care
4. Good Song
5. Love it.

A user selects a song and then selects the rating for that song. Furthermore a user is able to change ratings any time.

Implicit rating is done on assets that have not been rated in the past but that have been heard. Therefore, if a user registers to a stream and listens to a song, that song's rating should increase up to a certain threshold.

Stream rating is done without any direct user interaction. Every time a user selects a stream or leaves a stream a time stamp is set. The difference in the two time stamps decides whether the stream increases in rating.

Stream Suggestions

Delivery of stream suggestions to users is accomplished through three different techniques.

1. Stream content analysis.
2. User clustering.
3. Temporal analysis.

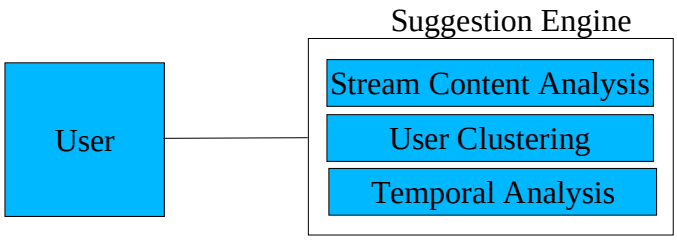


Figure 19 - SmartRadio Suggestion Engine

The first filtering technique, stream content analysis, is accomplished through analysis of each asset currently selected for streaming. An average is then computed on a per stream basis. The stream with the best average is selected as the basis for a suggestion to the user.

If no stream meets a certain static threshold this technique fails. On the whole, this is the most accurate suggestion because the user has rated items of the particular stream and the average rating has met the system threshold. However, this technique will most likely fail if either the user is a new users or the system is in a cold-start state.

If this technique fails, SmartRadio employs the second technique called user clustering. This is accomplished by computing the nearest neighbor to the current user and then performing the previous technique with this new user. To compute the nearest neighbor, SmartRadio uses the *mean squared difference* formula:

$$Mean\ Squared\ Difference_{uj} = \left(\frac{1}{|inCommon|}\right) \cdot \sum_{f \in inCommon} (u_f - j_f)^2$$

Figure 20 - Mean Squared Difference

This nearest neighbor collaborative filtering technique is often used in Recommender Systems. However it may still fail under the same circumstances as above.

Finally if both techniques mentioned above fail to retrieve a suggestion, the third technique, temporal analysis, is used. With temporal analysis, a stream is picked based on its popularity in a certain time frame. For example, the most popular stream closest to the current time is suggested to the user as the most likely stream to be preferred by that user.

Dynamic Customized Playlists

The second responsibility of the recommender system is to generate dynamic playlists based on user's preferences. This is done through 6 different steps:

1. Compute highest average asset from listening user set.
2. Prune 20% of listening users.
3. Increase rating weight for 'Hard Core' users.
4. Acquire assets from adjacent streams.
5. Choose random asset from playlist
6. Reset playlist.

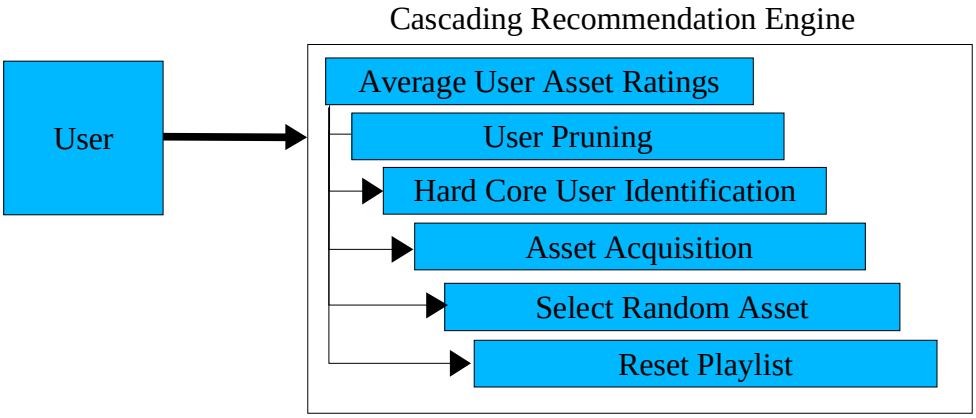


Figure 21 - Cascading Recommendation Engine

The recommendation engine is a cascading event model. That is, as one method fails the next is attempted in succession. Therefore, each method down the cascading model is less and less likely to be invoked.

The first step in dynamic content aggregation is average computation for assets in a stream. This is done so that the highest rated average can be retrieved. If no assets belonging to the playlist are above a static threshold, this step fails.

The second is step is to prune users who may be polluting this stream. These are users who do not belong in the current stream or who have negatively rated all remaining assets. Once a listener is pruned, a message is passed through the controller recommending the user to switch to another stream.

The third step in determining the next asset to queue for streaming is based on the concept of 'Hard Core' users. Hard core users are users who have been loyal to a particular stream. These users have more sway over the next asset to be played. Thus, SmartRadio scans all listeners for Hard Core listeners and increases these listeners overall rating weights.

The fourth step is asset acquisition. This step retrieves assets that meet the play threshold from adjacent streams. When a song is acquired it remains in the destination stream without being removed from the source stream.

If all the previous steps fail then SmartRadio plays a random asset. This is done to guarantee that all songs in a playlist get played at least once and to guarantee that the playlist does not enter a loop where only one song gets played.

Finally, this step resets the playlist and begins playing from the first song. At this stage the playlist has been played in its entirety.

3.2.0.2 SmartRadio View

The View layer in SmartRadio presents data to the user in two different fashions.

- Firstly it presents recommender data, system statistics and stream information to the user through a web browser interface. Recommender data should include suggested streams and asset ratings. System statistics include number of logins, favorite stream, number of online users, etc. Stream information includes content, which is currently being streamed through all streams, and the particular stream the user is registered with. This is accomplished through a combination of Servlet and JSP server side code.
- Secondly it presents the actual stream content to streamer through the stream connector.

Stream Connector

The stream connector is another view of the MVC design model. It is responsible for delivering audio content to the streamer. The streamer then distributes that content to the end user.

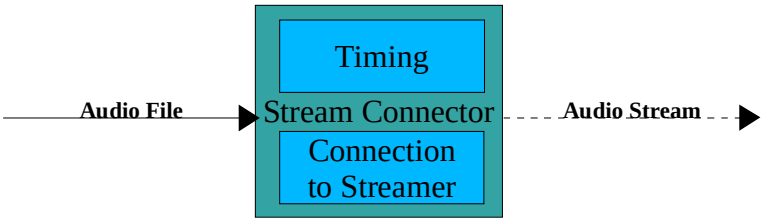


Figure 22 - Stream Connector

Keeping the stream connector and the streamer as different applications has many advantages. Amongst these advantages are:

- scalability

- modularity.

Modularity means that if over time SmartRadio evolves into a full multimedia streaming solution then the only components that need to be altered are the streamer and the stream connector. Furthermore, replacing the streamer with a different technology such as a unicast technology will not affect the rest of the system. In essence, as long as the protocol between the Stream Connector and the Streamer remains unchanged, both components can be completely rewritten independently of each other.

Scalability is important due to the high processing that is inherent in multimedia applications. If SmartRadio were to grow to support thousands of users listening to hundreds of simultaneous streams it would become overloaded. However, by splitting the streamer and the stream connector the processing needs can be reduced greatly.

Timing

Once the stream connector has an audio file from its input and has a connection to the streamer through its output it begins streaming the file. In order for an audio file to be played back correctly at the receiver it must be streamed at the same rate at which it was first sampled. For example a file sampled at 128bits/sec must be streamed at the same rate. If it is streamed too fast, the receiver's buffer will overflow causing the audio to skip. If it is streamed too slowly the receiver's buffer will empty and there will be no data to playback causing the audio to skip until the buffer fills up again.

Therefore the stream connector must provide a timing mechanism in which the stream rate will match the audio file's original sample rate. The timing mechanism works by retrieving a chunk of data from the audio file and then determining how much time that chunk of data represents.

Data Chunk = 16000 Bytes Sample Rate = 128 kbps 16000*8 = 128000bits 128000/128000 = 1 second. *There is 1 second of audio content in 16 Bytes of data at a data rate of 128bps.*

In this example the stream connector would delay streaming the next chunk of data for 1

second because it would take 1 second for the client to consume the previous chunk.

3.2.1 SmartRadio Data Layer

The persistence layer is arguably the most important layer of the three-layer architecture. SmartRadio uses a relational database server as its persistence layer. Because there are many database standards and database applications available today, SmartRadio design should be database-independent. This means that changing the SmartRadio persistence layer should easy and should not require any altering of the business logic. New relational databases should be integrated transparently without affecting existing application code because the persistence layer is decoupled from application logic. This can also help decouple programmers who write data layer code from those who write application layer code.

The key issue in picking a relational database is that of concurrency. The database must provide a means of preventing errors that occur when one user concurrently accesses data that is being accessed by another user. Concurrent access, particularly updates, can give rise to errors in a database. Therefore the database server should provide a locking mechanism by which the application can lock areas of the database so that access is only allowed to one user at a time. Furthermore, the database server should detect and act upon deadlock. Deadlock occurs when one user transaction has exclusive access to a resource such as an SQL table and is waiting for a resource that is held exclusively by another user transaction. A database server should detect such serious conditions and remedy.

The relational database used by SmartRadio is mysql. Mysql supports transactions with the InnoDB and BDB Transactional table handlers.

3.3 The Streamer Tier

The streamer is the final tier of the three-tier model used by SmartRadio. It is responsible for streaming the media content to the client. Many media streamers in use today use unicast as the network base over which to stream multimedia content. However, with unicast, the bandwidth demands placed on the network are to great, causing unicast to scale

poorly in large and very busy streaming servers.

To solve the problem of scalability SmartRadio uses multicast as the base over which it services clients. Multicast is efficient because it grows with the number of streams currently in use and not with the number of users listening as is the case with unicast. Multicast transmission runs on top of the UDP layer and thus does not provide any of the features readily available under TCP.

To solve the transport layer problems of UDP, SmartRadio uses the RTP transport protocol to provide delivery of audio packets. RTP provides end-to-end delivery services to packets with real-time characteristics. These services include;

1. Payload type identification
2. Sequence numbering
3. Time Stamping
4. Delivery Monitoring.

Furthermore RTP supports data transfer to multiple destinations using multicast distribution.

Even though the benefits of multicast are clear, most networks in use today do not support it. Therefore, in order for SmartRadio to be a deployable application it must also support unicast as a last measure service to those that do not support multicast.

Figure 23 depicts the streamer behavior with both unicast and multicast clients. In this figure two unicast clients listen to the same stream while three multicast clients listen to the same stream. It is important to note that multicast is a push technology and therefore begins streaming to a 'channel' the moment data is available. A channel basically consists of a port and multicast address. The channel on which the stream is broadcast needs to be communicated to clients in an out-of-bands means. SmartRadio communicates the channel information to its listeners through the SmartRadio web site.

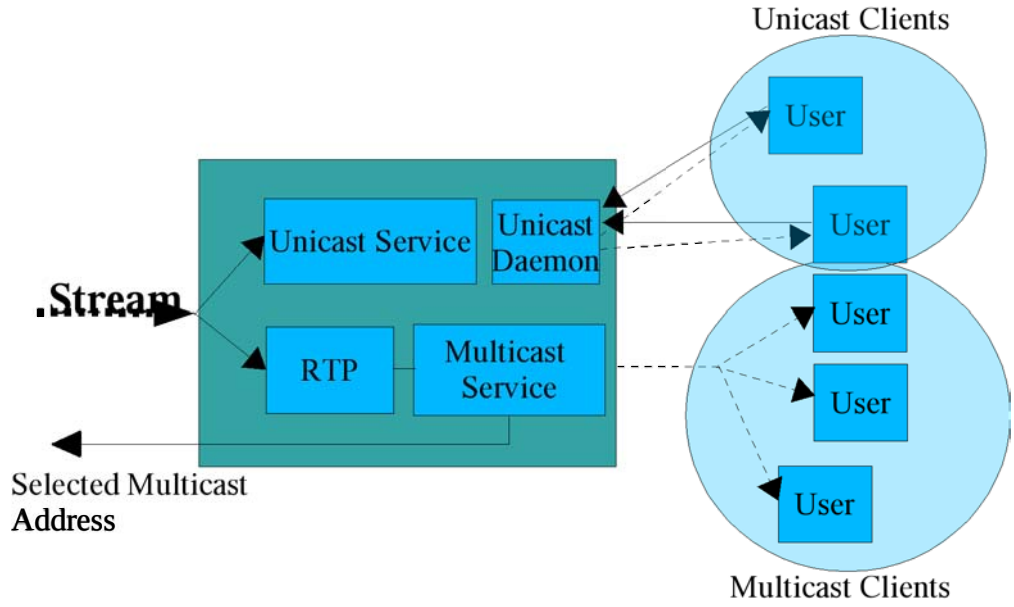


Figure 23 - SmartRadio Streamer

The unicast clients need to first contact the streamer and ask for the information. Once the GET query is received, the stream begins streaming to the client. Each client gets a copy of the same stream. In other words, unicast connection requests need to be received by the server in order to be processed.

With multicast, the stream is received by the streamer encapsulated in an RTP packet and streamed over the multicast network. Clients register to the multicast group on the edges of the network.

3.4 SmartRadio System Summary

This chapter reviewed the design and analysis of SmartRadio. Figure 24 Illustrates the entire system architecture bar unicast clients.

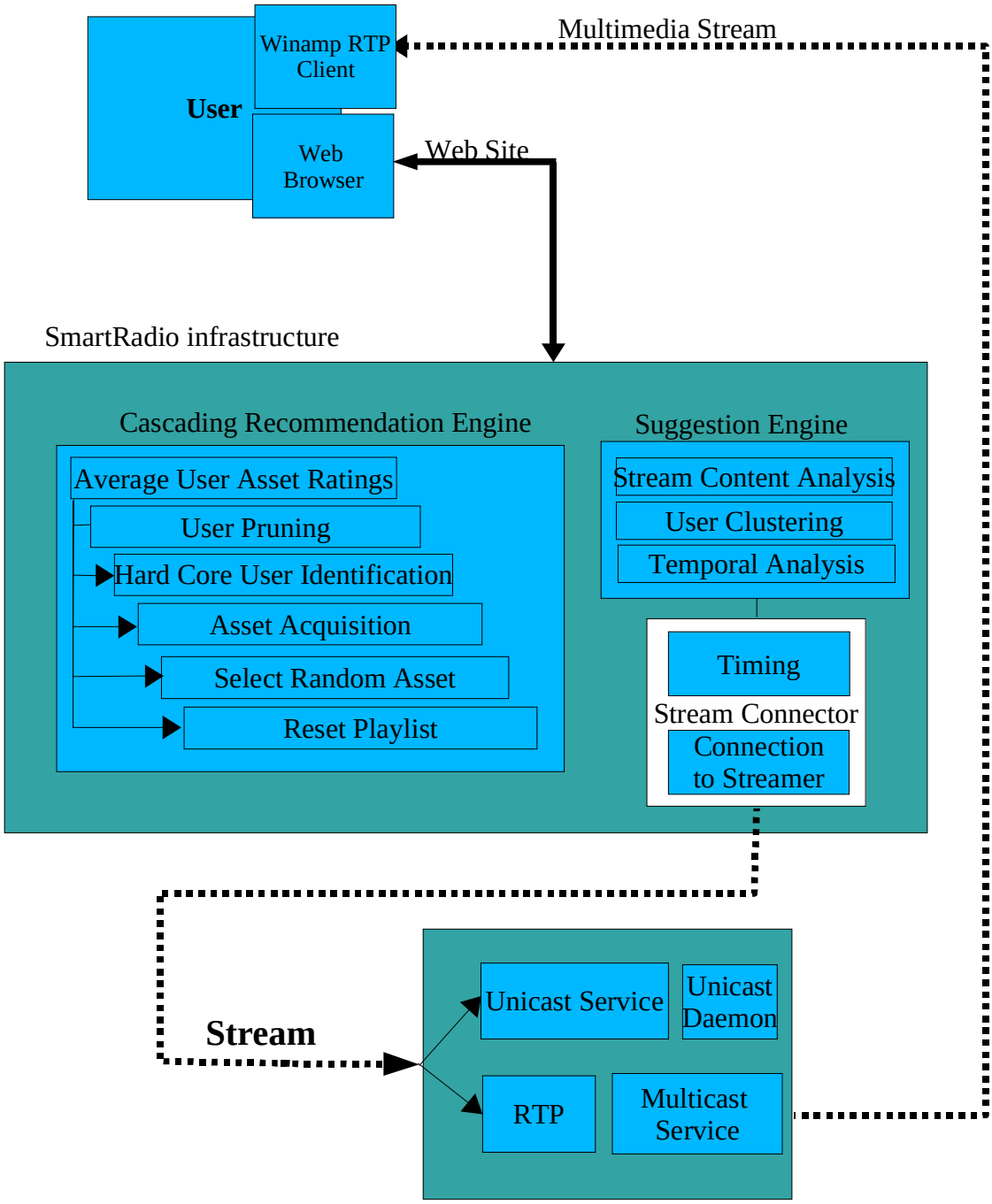


Figure 24 - SmartRadio

4 Implementation

Like the design, the implementation of SmartRadio is broken down into three major components.

1. Implementation of the user side RTP streaming client.
2. Implementation of the streamer.
3. Implementation of the Web site.

4.1 User Side RTP Streaming Client

Nullsoft's Winamp is the most popular audio player for Microsoft Windows. However, Winamp does not natively support RTP multicast streaming so a plugin is needed. A Winamp client plugin is available free of charge from Live Networks.

The input plugin: in_rtp.dll allows Winamp to receive RTP-based MP3 streams directly, without the need for a separate multicast receiver 'helper application'. In_rtp.dll works only with Winamp versions 2.6 through 2.81, earlier version have a bug that can prevent this plugin from working properly. Also, the plugin will not work with "Winamp3".

Finally, winamp has native support for unicast streaming. Therefore, if the RTP client is unavailable or the Winamp version is incompatible the user may still listen to SmartRadio streams using unicast based networking.

Under Linux, a variety of RTP multicast streaming clients exist. Freeamp version 2.1.1 is an Open Source effort to build a digital audio player. It has native support for RTP media streaming as well as a host of other features. Like Winamp, Freeamp comes with native support for unicast streaming. Zinf is another audio player for Linux and Win32. It supports MP3, Ogg/Vorbis, WAV and Audio CD playback, SHOUTcast/Icecast HTTP streaming, RTP streaming.

4.2 Multicast RTP Streamer

The multicast RTP Streamer is developed completely under Linux version 2.4.18-3 on a x86 Pentium III 700Mhz computer.

As emphasized in the previous chapter, SmartRadio must implement unicast streaming in order to support the clients who are connected through networks that do not support multicast. There are numerous unicast streaming solutions available today, for example RealServer, Shoutcast, Icecast.

RealServer

The complete version of RealServer (RealServer Plus) for Linux is commercially available from RealNetworks.com. RealServer Plus is industrial-strength software and certainly provides everything for network broadcasting. However, it's a proprietary closed source Internet broadcasting solution with a fairly hefty price tag. Which means that its source code is not available for developers to modify or extend. Furthermore, it's not an all-in-one solution, RealProducer content-creation software is also needed in order to make RealAudio files from pre-existing sound files or prepare live netcasts in RealAudio streams. These factors make Realserver a poor solution for SmartRadio.

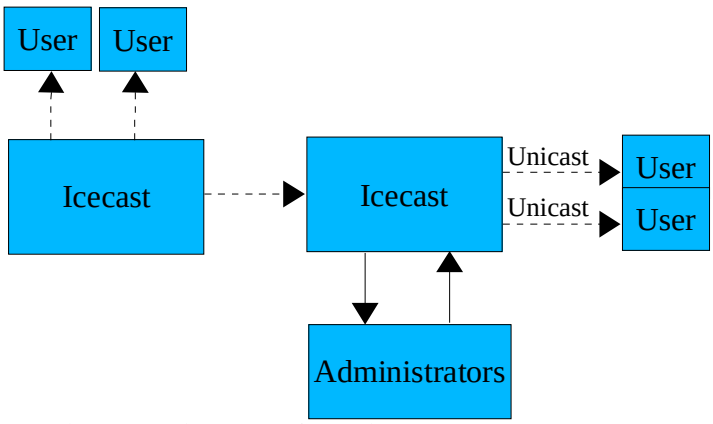
SHOUTcast

SHOUTcast is Nullsoft's Free Winamp-based distributed streaming audio system. It has native support for the MP3 audio format. However, shoutcast is not open source and does not have support for multicast RTP streaming which also makes it a poor choice for SmartRadio.

Icecast

Icecast is a free open source streaming server licensed under the GPL. It is easy to install, configure, and use, making it possible for anyone with sufficient software and hardware resources to become an instant Internet broadcaster. Icecast 1 supports streaming MP3s and M3U playlists, and version 2 supports streaming Vorbis files. Icecast 2 is the first streaming Vorbis server.

Icecast is entirely written in C. Essentially it works by accepting incoming stream connections and then serving as a unicast stream server to other clients. Additionally Icecast supports external console administration. Administrators can open a remote console telnet connection and through it administer Icecast streams.



Drawing 1 - Basic Icecast Operation

Icecast natively supports hierarchical based stream sharing. This can be used to make Icecast scale when the system experiences high usage. With hierarchical connections, one Icecast server connects to another Icecast server running elsewhere and rebroadcasts that server's streams.

The combination of open source GPL licensing and support for a variety of audio formats makes Icecast the ideal unicast solution for SmartRadio. However, the current version of Icecast version 1.3.12 has no native support for Multicast. Therefore it will need to be extended to support RTP multicast streaming.

RTP Software Development Kit

In order to extend Icecast's functionality to support Multicast RTP streaming a new set of libraries have to be used. Instead of writing a specific implementation from scratch, Live Networks RTP libraries are used. These libraries have already been used in real RTP-based applications, and are well-suited for use within SmartRadio.

The libraries are written in C++. The code includes an implementation of RTCP, and can easily be extended (via subclassing) to support new RTP payload types.

UsageEnvironment

The "UsageEnvironment" and "TaskScheduler" classes (header file: "include/UsageEnvironment.hh") are used for scheduling deferred events, for assigning handlers for asynchronous read events, and for outputting error/warning messages. Also, the "HashTable" class (header file: "include/HashTable.hh") defines the interface to a generic hash table, used by the rest of the code. These are all abstract base classes; they must be subclassed for use in an implementation. These subclasses can exploit the particular properties of the environment in which the program will run.

Groupsock

The classes in this library encapsulate network interfaces and sockets. In particular, the "Groupsock" class (header file: "include/Groupsock.hh") encapsulates a socket for sending (and/or receiving) multicast datagrams.

liveMedia

This library defines a class hierarchy - rooted in the "Media" class (header file: "include/Media.hh") - for a variety of streaming media types and codecs. See Appendix B for a list of supported Media.

BasicUsageEnvironment

This library defines one concrete implementation (i.e., subclasses) of the "UsageEnvironment" classes, for use in simple, console applications. Read events and delayed operations are handled using a select() loop.

4.2.0.1 SmartRadio Icecast RTP Implementation

In order to merge Icecast and the RTP software kit, Icecast has to be converted into C++. Once the program supports the object oriented paradigm, the RTP SDK could be merged with Icecast.

The two major modifications that need to be made to Icecast are:

- Icecast protocol modification
- Multicast RTP Transmission addition.

4.2.0.2 Icecast Protocol Implementation

As described in the previous chapter, the Icecast streamer application must communicate the channel used for multicast streaming back to the stream connector. To implement this the Icecast protocol needs to be altered.
Currently the Icecast protocol supports two connection types:

1. The source: Is the incoming stream which Icecast uses to service requests.
2. The client: The clients who connect to Icecast and receive streams.

SmartRadio's stream connector connects to Icecast as a source. The stream supplied by the stream connector is redistributed to streaming clients. However, the multicast address over which Icecast pushes content is unknown to the stream connector. Therefore, the source protocol has to be modified to communicate this information to the stream connector.

The client protocol, on the other hand, remains unchanged to be compatible with all regular unicast streaming clients.

The only information currently supplied to the source by Icecast is success or failure of the stream connection. This must be modified to add the Multicast address and port over which the multicast transmission is to take place.

Source	Modified Icecast Protocol
SOURCE PASSWORD / mountPoint\n	OK MULTICASTADDRESS:PORT\n
{ IcyHeader }	...
{ Stream Date }	...

Table 3 - Icecast Protocol

Each source connected to Icecast receives its own Multicast address over which to stream. Address are allocated from a a linked list class called MulticastAddressList. Once an address has been allocated it is not given to any other sources until the original source for which it was allocated gets torn down. A multicast address is composed of an IP and a Port and it is stored in the Maddress class. The Maddress class, along with the actual source, gets passed to the multicast streamer thread which is described in the next subsection.

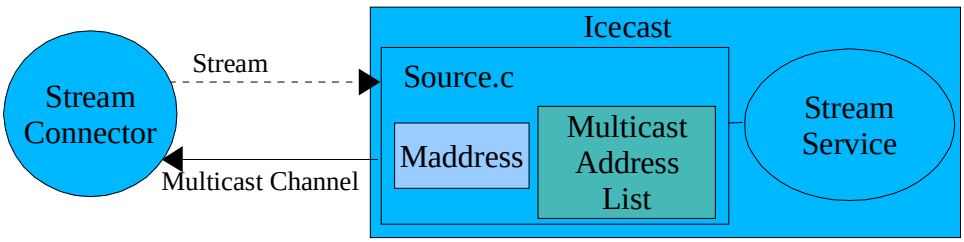
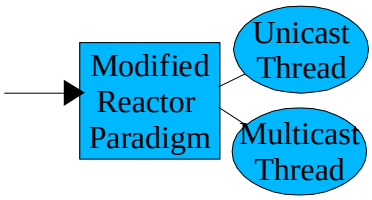


Figure 25 - Multicast Channel Transmission

4.2.0.3 Multicast RTP Transmission

Icecast currently spawns a new thread to handle new incoming connection. This paradigm is called the Reactor paradigm. The reactor paradigm needs to be modified to spawn an additional thread per connection which will be responsible for multicast RTP packets.



Drawing 2 - Modified Reactor Paradigm

The multicast thread spawned at the same time as the unicast thread contains all RTP and Multicast streaming code. The header file for this thread looks as follows:

```
void *mcast_init(void *);
char *normalizeAddress(unsigned char *);
void afterPlaying(void* sessionState);

struct sessionStateSt {
    FramedSource* source;
    RTPSink* sink;
    RTCPInstance* rtcpInstance;
    Groupsock* rtpGroupsock;
    Groupsock* rtcpGroupsock;
    MAddress *address;
};
```

The session State struct is used to store current multicast session information. This is used to allow Icecast to shutdown the session gracefully.

mcast_init is called when the thread is started, mcast_init expects an argument of type 'mcontainer_t'. This struct is used to store the multicast address over which the connection will be streamed. Furthermore, the source is stored here. The source information contains the mount-point over which the source is connected. A mount-point in Icecast is a naming abstraction used to differentiate between sources.

```
typedef struct mcontainerSt
{
    MAddress *address;
    source_t *source;
} mcontainer_t;
```

Figure 22 illustrates the whole Icecast RTP Multicast streamer.

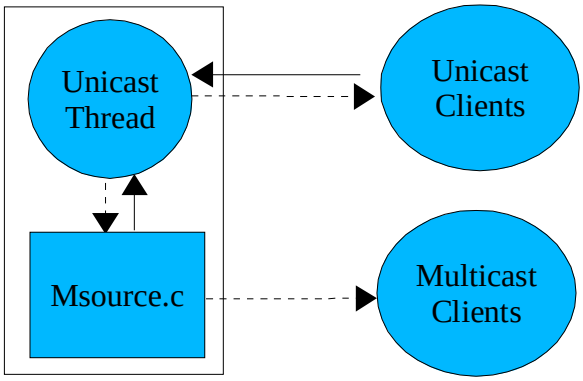


Figure 26 - Icecast RTP Multicast Streaming

4.3 SmartRadio Website Implementation

SmartRadio's fundamental functionality is to stream audio content to registered clients. This function cannot be performed with audio content being present on the system in the first place. Therefore the first implementation issue is the loading of audio content into the persistent layer.

4.3.1 MP3FileLoader

The MP3FileLoader application is an independent application developed outside the scope of both the website and the streamer. Although developed on Linux, it is written in Java and can used on any system.

The MP3FileLoader application takes advantage of the ID3 file tags that are part of each MP3 file. It extracts ID3 information, namely:

- Filename
- Title
- Artist
- Genre

and stores it into the database. ID3 tag information is accessed through a third party package written by Jens Vonderheide <jens@vdheide.de). All files that have incorrect ID3 information or that do not contain all the relevant fields described above are not added to the database. See appendix A for supported genre types.

4.3.2 SmartRadio Website

Recall the design figure from the previous chapter.

All development took place on the application or middle layer. The persistence layer consists of the mysql relational database server. This server supports transactions and is fully supported by the Java API.

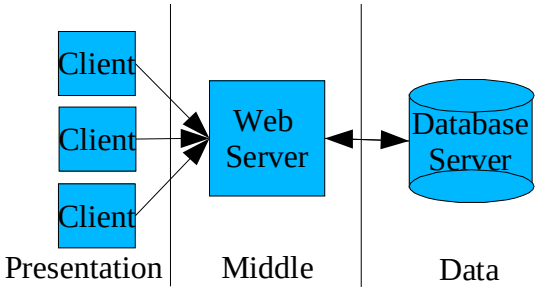


Figure 27 - Three Tier Model

4.3.2.1 Presentation Layer

Typically presentation layer implementation causes a lot of code duplication. Every page created has many of the same features such as background. Furthermore some of the functionality should be present on every page, such as a hit counter or the name of current active streams.

Minimizing code duplication would allow for diminished maintenance. SmartRadio uses the template functionality that is part of Struts in order to reduce code duplication.

The "struts-template" tag library contains tags that are useful in creating dynamic JSP templates for pages which share a common format. These templates are best used when it is likely that a layout shared by several pages in your application will change. The functionality provided by these tags is similar to what can be achieved using standard JSP include directive, but are dynamic rather than static.

With the struts template library, one template is defined as the master template.

“tempate.jsp”

All other pages then use that template and simply fill in their custom information.

```
<template:insert template='/template.jsp'>
<template:put name='refresh' content='<META HTTP-
EQUIV="Refresh" CONTENT="100; URL=mainMenu.jsp">'
direct='true' />
<template:put name='title' content='Welcome to SmartRadio
BETA' direct='true' />
<template:put
name='body'content='mainMenu/mainMenu_body.jsp' />
<template:put
name='user'content='mainMenu/mainMenu_user.jsp' />
<template:put name='system'
content='mainMenu/mainMenu_system.jsp' />
</template:insert>
```

4.3.2.2 Application layer implementation

As described in the previous chapter, the application layer was developed with the aid of Struts. Each action performed by the system in response to a user action is stored in action classes. As depicted in Figure 28 a user can:

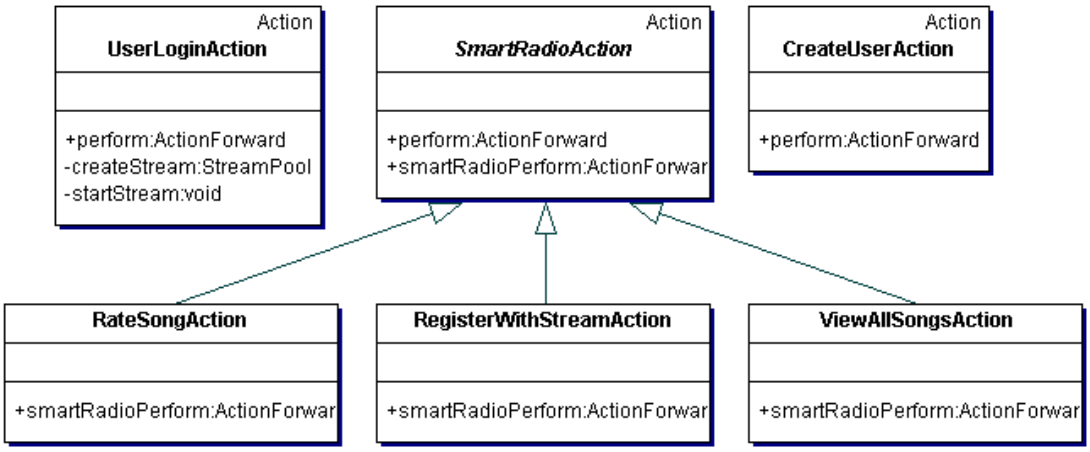


Figure 28 - Action Classes inheritance graph

1. Create a new user
2. Login
3. Rate an Asset
4. Register with a stream
5. View all queued assets from a specific stream

The super class 'Action' is provided by Struts. The super class 'SmartRadioAction' is used by classes that wish to guarantee that a user is logged in before allowing access to their functionality. This way the internal mechanisms of SmartRadio are protected from users who have not logged in.

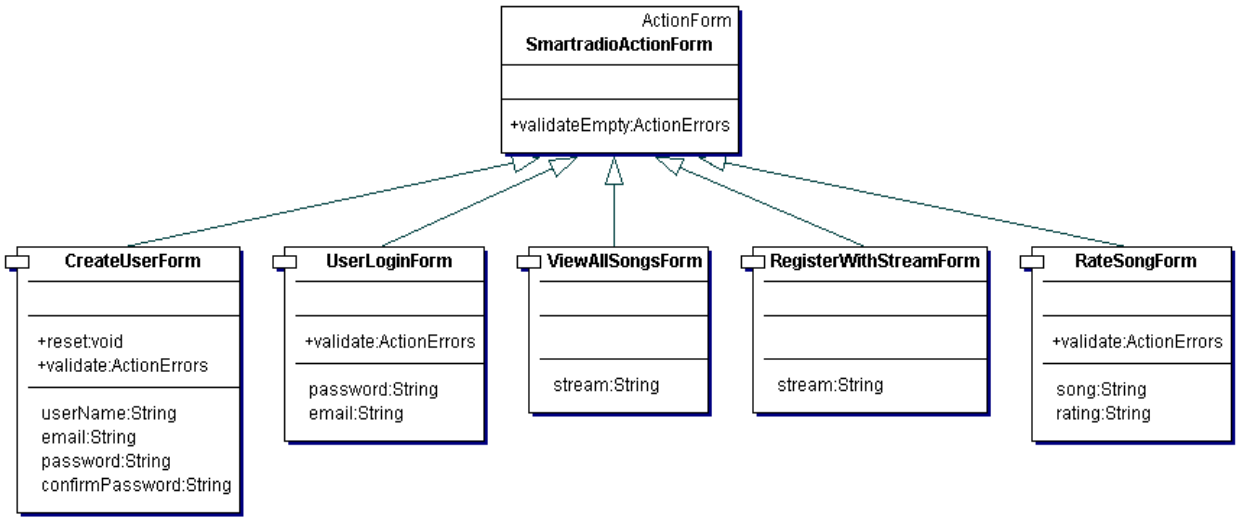


Figure 29 - SmartRadio Action Forms

A user is expected to send information to SmartRadio through a form. Forms in SmartRadio are classes whose properties are to be filled by Struts through introspection. Figure XXX shows all the forms developed for SmartRadio.

The super class 'SmartradioActionForm' is implemented to filter out any inconsistencies with form data. These could include forgetting to put a '@' in an email field or leaving a field blank.

Streams

Recall that streams are loaded upon the login of the first user to the system. They must remain in memory and accessible by all users. Furthermore they must be continually executing so serializing and storing them in the persistence layer would not be appropriate. Thus upon creation streams are stored in the servlet context. According to the Java Servlet Specification: “A servlet can bind an object attribute into the context by name. Any attribute bound into a context is available to any other servlet that is part of the same web application” [JSS]. To bind and unbind objects from the servlet context the following methods are user:

- setAttribute
- getAttribute
- getAttributeNames

- removeAttribute

SmartRadio has a 'StreamPool' in which all streams are stored. Table 4 shows the process of creating and storing new streams within the streampool.

```
streamPool = StreamPool.getInstance();
startStream("rock", errors, streamPool);
startStream("Oldies", errors, streamPool);
startStream("Hip-Hop", errors, streamPool);
startStream("Alternative", errors, streamPool);
startStream("Pop", errors, streamPool);
startStream("RnB", errors, streamPool);
context.setAttribute("streamPool", streamPool);
```

Table 4 - Adding to StreamPool

When a stream is terminated or malfunctioning, the StreamPool container must be notified and remove that stream from the list of active streams. This is accomplished through event oriented programming.

Stream Event Model

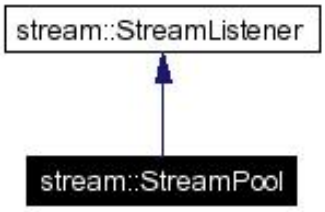


Figure 30 - StreamPool Class

Streams are responsible for managing playlists and streaming audio. Furthermore they are responsible for notifying related classes of their state. A stream can either be streaming or stopped. It is stopped when there are no listeners. In the HTTP protocol, there is no explicit termination signal when a client is no longer active. This means that the only mechanism that can be used to indicate when a client is no longer active is a time-out period. In order to keep an active listener from timing-out the HTTP meta refresh tag is used:

```
<META      HTTP-EQUIV="Refresh"      CONTENT="100;
URL=mainMenu.jsp">
```

Table 5 - Meta Refresh Tag

To begin streaming, the Stream class uses the SmartStream package.

The SmartStream package is the heart of the Stream Connector explained in the previous chapter. It is responsible for connecting to the streamer and streaming the audio content. Furthermore it is responsible for the timing specifications of streamed media. This class is also oriented around events. When a stream fails or a connection to the streamer is dropped, SmartRadio passes an event to the Stream class, which in turn passes an event to the StreamPool class. Table 1 lists all the events used in SmartRadio.

HTTPDataSink	***SmartStream***	***Stream***

`DataSinkMAddressEvent`	`SmartStreamAddressEvent`
	`SmartStreamEndOfMediaEvent`
	`SmartStreamErrorEvent`
	`SmartStreamEvent`
	`SmartStreamStopEvent`
		StreamEvent

Table 6 - Stream Event Model

All SmartStream events extends from 'SmartStreamEvent.' Figure 27 shows the inheritance graph for all SmartStreamEvents.

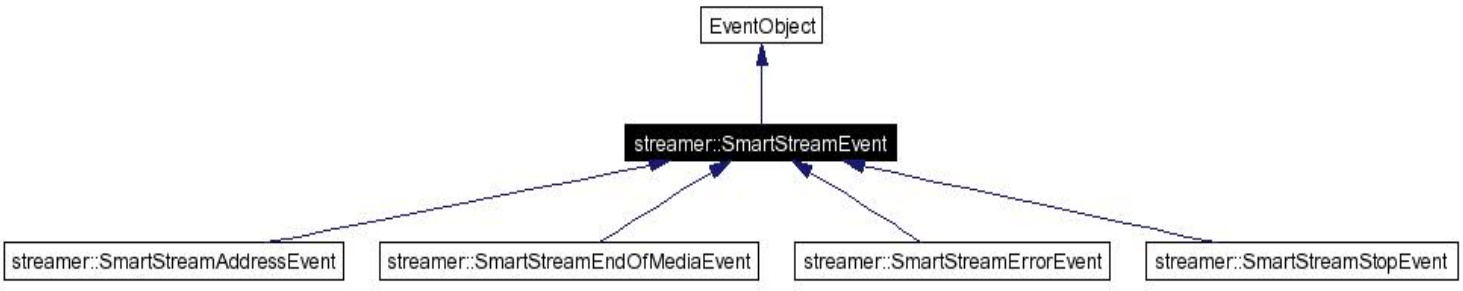


Figure 31 - SmartStream Events

SmartStream: Stream Connector

The SmartStream package implements the Stream Connector explained in the previous chapter. In order to allow for new media to be inserted easily into the SmartRadio model, SmartStream uses the Java Media Framework (JMF) as its streaming base.

The JMF is an application programming interface for incorporating time-based media into Java applications and applets. JMF 2.0 API, which is used in this SmartRadio implementation, provides support for capturing and storing media data, controlling the type of processing that is performed during playback, and performing custom processing on media data stream. But more importantly, JMF 2.0 defines a plug-in API that enables SmartRadio to easily customize and extends JMF functionality. Finally, JMF supports RTP streaming on its own.

If in the future it is decided that unicast compatibility is unimportant, then the streamer can be removed from the SmartRadio model and JMF can be used to multicast RTP media packets.

Currently in SmartRadio, the main JMF feature used is: Access to raw media data. JMF supports a large number of media data, such as, AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF and WAV. Figure 31 demonstrates the overall SmartStream class.

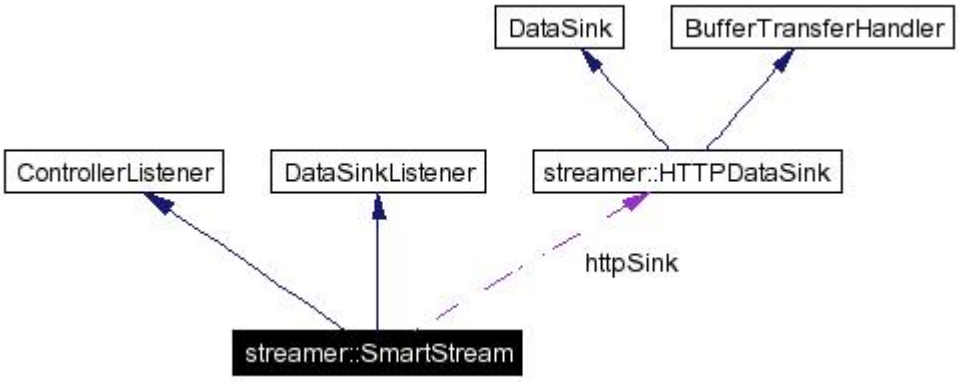


Figure 32 - SmartStream Class

The Controller Listener and the DataSinkListener are used for event monitoring. HTTPDataSink is used to connect to the Icecast streamer and deliver appropriate timing. The timing is shown in the following table.

```
try {
    this.wait(((readBuffer.getLength() * 8) /
bitRate));
} catch (InterruptedException e) {
    System.err.println(e);
    sendEvent(new DataSinkErrorEvent(this,
e.getMessage()));
    stop();
}
```

Table 7 - Timing Calculations

Play Lists

All streams contain playlists which are ultimately the content streamed to the user. In essence the SmartPlaylist class employs the recommender system to generate dynamic content. Playlist is the super class and is responsible for audio storage.

Figure 33 - SmartPlayList inheritance graph

SmartPlayList

As discussed in the design chapter, dynamic content is built using the following steps:

1. Compute highest average asset from listening user set.
2. Prune 20% of listening users.
3. Increase rating weight for 'Hard Core' users.
4. Acquire assets from adjacent streams.
5. Choose random asset from playlist
6. Reset playlist.

Each step represents a line of the loop bellow.

```
while (bestAverageRating < threshold)
        if (!prunedUsers)
                prunedUsers=true;
        else if (!normalizeUserStreamWeight)
                normalizeUserStreamWeight   =
true;
        else if (!acquiredFiles)
                acquiredFiles=true;
        else if (TotalUsers < 1)
                PICK RANDOM SONG
        else
                RESET STREAM
```

Table 8 - SmartPlayList Loop

Another feature of the SmartRadio SmartPlayList class is automatic asset rating. This is done automatically without direct user interaction. The idea behind this is that if a user has not explicitly rated any assets but has listened to them, then he must prefer them over others. In this way, each time a song is listened in its entirety it gets an automatic user rating. In order to accomplish this, SmartRadio marks all the users listening to a stream in between each song. If a user is registered on a stream before a song and directly after then it is assumed he listened to the whole song. In this case the user's rating for that asset increases up to a certain threshold.

Both the smartplaylist and the suggestion generator depend on a set of constants set in the Constants class. These constants are listen in table 2.

```
public final static int STREAMTHRESHOLD = 10;
public final static double
USERPRUNINGPERCENTAGE=20;
public final static double MINIMUMPLAYTHRESHOLD
= 2.25;
public final static double
STREAMRATINGTHRESHOLD = 2.25;
public final static String LISTENEDRATING="2";
```

Table 9 - List of Constants

The STREAMTHRESHOLD is the amount of minutes a user has to be listening to a stream before their rating of that stream increases.

The USERPRUNINGPERCENTAGE is the amount of users to prune from the set of listeners as described in the SmartPlayList step 2.

The MINIMUMPLAYTHRESHOLD is the minimum rating an asset must have if it is to be queued for play. This value is used in the SmartPlayList loop.

The STREAMRATINGTHESHOLD is the minimum rating a stream must have before it is suggested to a user.

The LISTENEDRATING is the rating to allocate an asset after a user has heard it but has never rated it in the past. This is the threshold used by SmartRadio's inference rating model.

Recommender System

Figure 34 depicts the whole recommender system architecture.

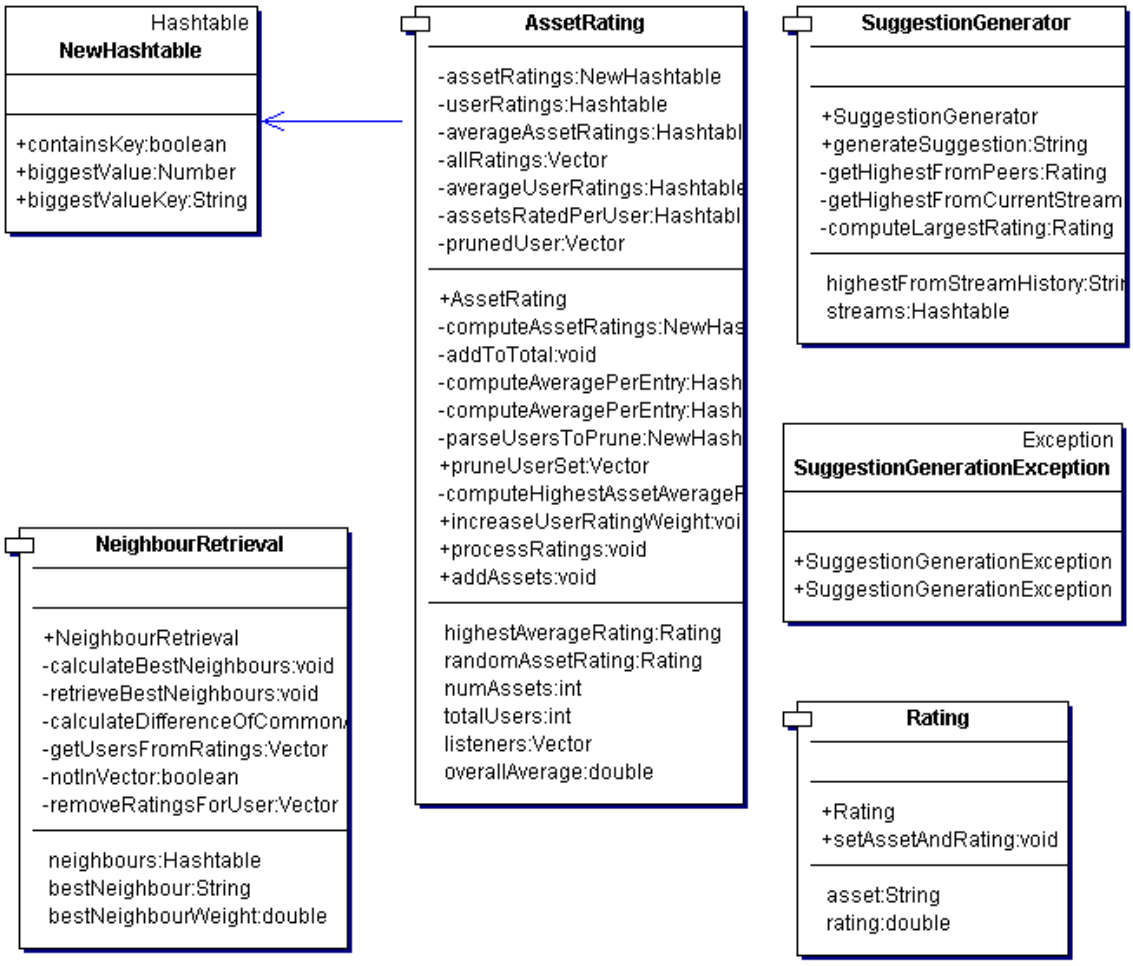


Figure 34 - SmartRadio Recommender System

The steps required to generate a suggestion to each user are:

2. Stream content analysis.

3. User clustering.

4. Temporal analysis.

The pseudo code used to accomplish these three steps is shown in Table 10.

```
if (getHighestFromCurrentStreams >= STREAMRATINGTHRESHOLD)
    return getHighestFromCurrentStreams;
if (getHighestFromPeers >= STREAMRATINGTHRESHOLD)
     return getHighestFromPeers;
return getHighestFromStreamHistory();
```

Table 10 - Recommender Pseudo Code

4.3.2.3 The persistence layer

SmartRadio has to consistently manage information that is to remain persistent. To accomplish this SmartRadio binds all objects that are to remain persistent to a relational database. The relational database used it MYSQL 3.23.51 for x86-Linux. The persistence layer is developed in such a way as to separate the database layer and the application layer as much as possible.

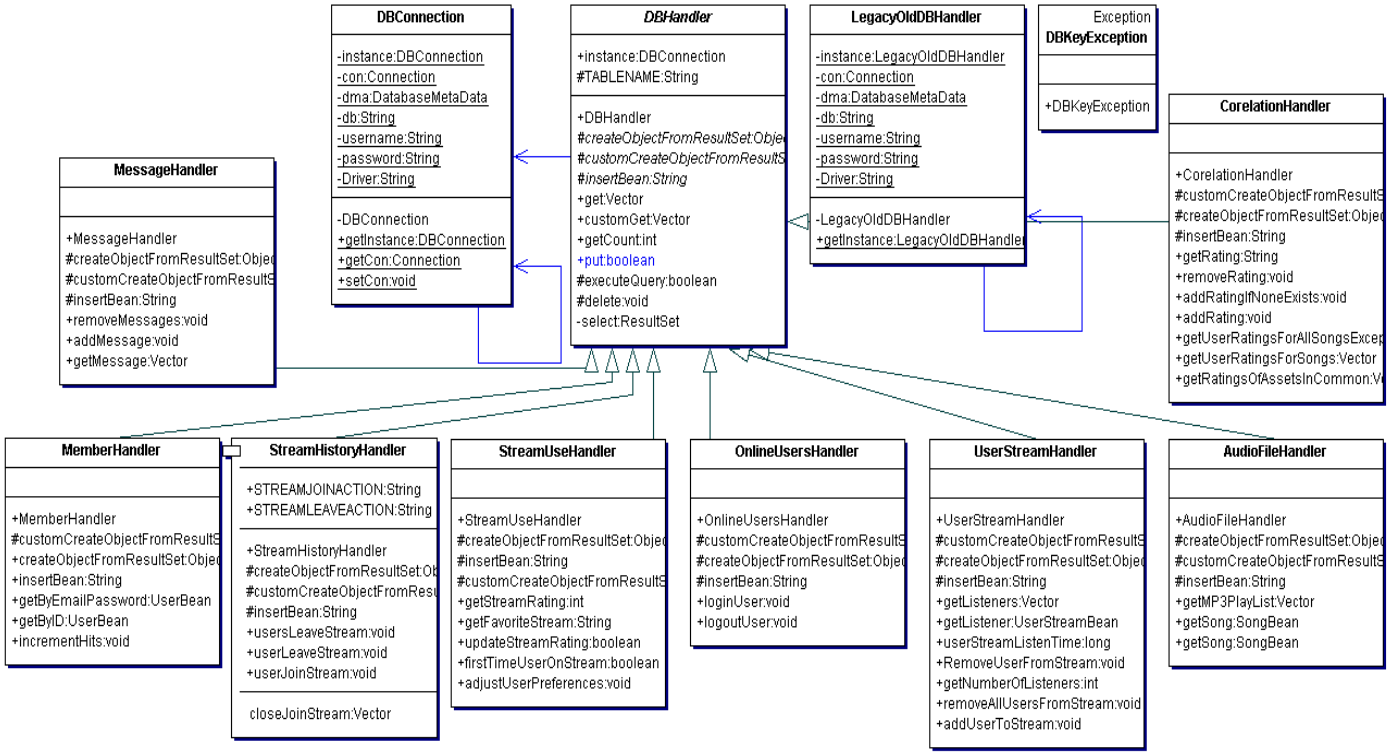


Figure 35 - Database Handler

In order to separate the persistence layer from the application layer, several abstraction layers were used. These abstraction layers are identified in Figure 35. If the persistence

layer is changed in any way, the only modification required to SmartRadio would to the individual bean handlers. Implementing transactions on top of this model is relatively straight forward. A new transaction class should be implemented which extends from the current DatabaseHandler class.

4.4 Summary

This chapter analyzed and explained the overall implementation of SmartRadio. Specifically it look at all three major components that make up the overall system.

- The streaming client on the client side.
- The streamer
- The Website

5 Evaluation

As with the design and implementation section, the evaluation chapter is broken down into three parts.

- The user side
- The web side
- The streamer side

This chapter evaluates all three sections independently and then finally evaluates the system as a whole. Evaluation consists of describing the actual functional and non functional parts of the system as well as possible improvements and future work. All evaluations are based on the original requirement which is reiterated here:

To create a scalable streaming application over the current IP multicast infrastructure and then using a Recommender Systems to suggest and customized, dynamic multimedia content for transmission over this medium.

5.1 The user side

Due to the nature of this thesis, most functionality can only be measured on the user side. Recommender systems are meant to provide desirable recommendations to the end user and only the end user can make a fair judgment of the their accuracy.

The user side is broken down into two parts:

- Stream client application
- Web client interface

5.1.1 Stream client Application

As iterated in the implementation section, the stream client used for SmartRadio can be either Winamp with the RTP plugin, or another RTP client streaming application.

Figure 36 - Winamp Audio client

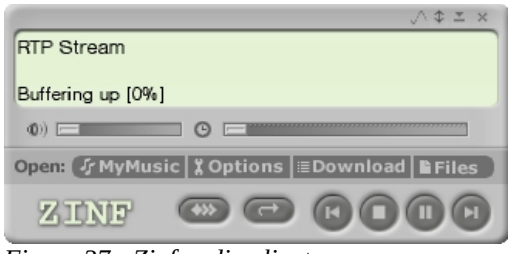


Figure 37 - Zinf audio client

There are many other possible clients that would be compatible with SmartRadio, these two were both extensively used and appeared to be very stable. Winamp seems to react negatively when changing from one multicast channel to another. In this case the stream appears to jitter terribly. After testing it was established that the stream was not the problem. Rather, Winamp has a problem which requires it to be restarted when switching from one multicast channel to another.

Although skipping is not an issue while testing on a LAN, it may become an issue when the source and the receiver are separated by many networks. In this case the payload of the RTP packet could be replaced from raw MP3 frames to a new MP3 Application Data Unit (ADU). RFC 3119[RFC 3119] describes a new RTP payload format for MPEG 1 or 2, layer III audio. This format is more resilient to packet loss than the format described in RFC 2250[RFC 2250]. Following the principle of "Application Layer Framing", packet boundaries in this new format correspond to MP3 ADUs, which - unlike in layer I or II - are not aligned on MPEG frame boundaries.

While the RTP payload format defined in RFC 2250[RFC 2250] is generally applicable to all forms of MPEG audio or video, it is sub-optimal for MP3. The reason for this is that an MP3 frame is not a true "Application Data Unit" - it contains a back-pointer to data in earlier frames, and so cannot be decoded independently of these earlier frames. Because RFC 2250[RFC 2250] defines that packet boundaries coincide with frame boundaries, it handles packet loss inefficiently when carrying MP3 data. The loss of an MP3 frame will render some data in previous frames useless, even if they are received without loss.

The RTP payload format described in RFC 3119[RFC 3119] uses a data-preserving rearrangement of the original MPEG frames, so that packet boundaries coincide with true MP3 "Application Data Units", which can also be rearranged in an interleaving pattern.

5.1.2 Web client Interface

The web client interface is used by SmartRadio to interact with its members. The priorities of any web user interface are ease of use and speed. Thus the SmartRadio user interface is primarily built for ease of use and speed. There are no superfluous graphics and the table design is very simple.

Figure 38 demonstrates a typical web page. It is designed to fit and look good on screens with a resolution of 800x600 or higher.

Figure 38 - SmartRadio Web Page

The current user interface displays all the contents of each genre's playlist in no particular order. It does not have any mechanism to display the current file being streamed or the queue of next few songs. Thus, the user interface could be improved by adding an option to rate the current playing song. Furthermore, the system could be improved by displaying the next few songs to be streamed. This, however, could not be an absolute list because

SmartRadio calculates the best song to play at the end of each song. It would therefore be impossible to predict within 100% accuracy the next 5 songs to be streamed.

Another major drawback of the SmartRadio design is the dependency on two separate applications. There is no way to start one application from another. As Figure 39 illustrates, a user must independently start their own streaming client to listen to a channel. There are two possible alternatives to the current SmartRadio user interface design.

1. Embed a multicast RTP streaming client within the website.
   This option is currently not possible.
2. Write a new plugin for winamp which would remove the need for the Web site interface all together. This would effectively combine both the streaming client and the web site. However, it would only be compatible with winamp and Microsoft Windows.

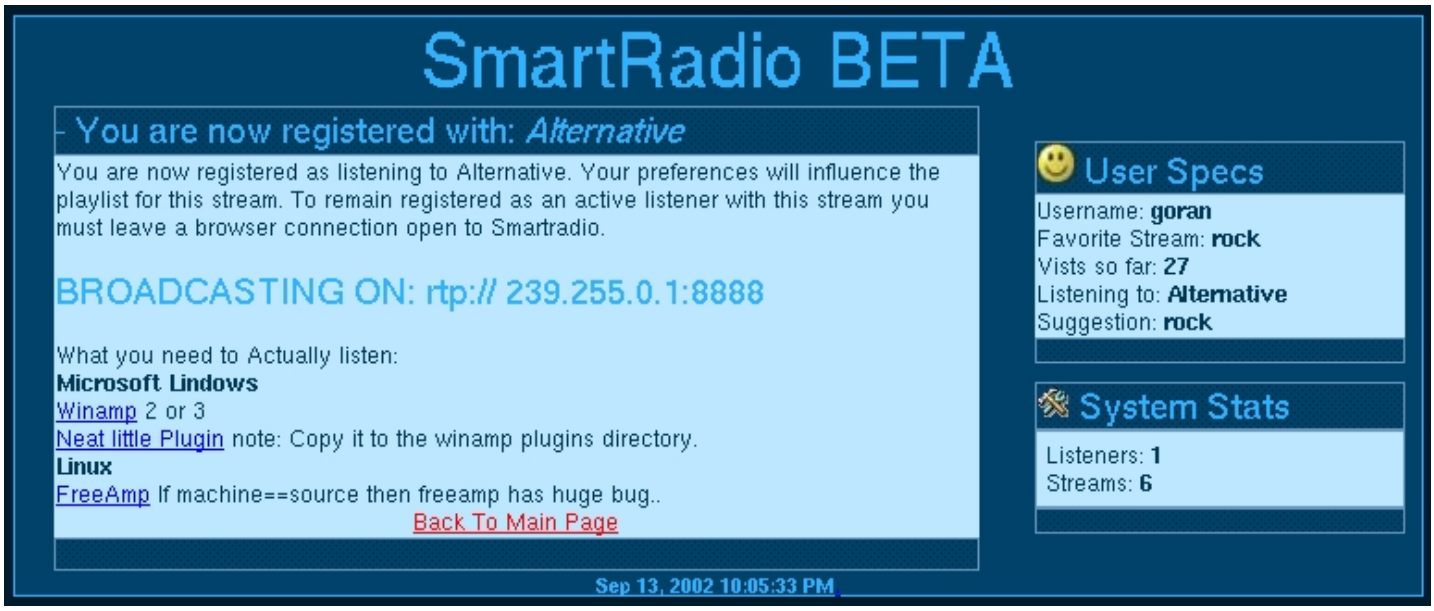


Figure 39 - Channel Registration Interface

5.2 The Web Side

The web side currently implements the recommender system and the stream connector. Although the design has been well implemented, there are some improvements possible.

As mentioned in previous chapters, all streams get initialized when the first user logs in. The major drawbacks of this are twofold:

1. Firstly, new assets cannot be added to a running system. Therefore a new version of SmartRadio could include a way for assets to be added to already initialized streams. This would involve creating a new step in the SmartPlayList class which acquires all new assets from the database before picking the next song to stream.
2. Secondly, new streams cannot be added to a running system. This issue would be more tricky to resolve. The system could perhaps periodically verify that there are enough new assets of a certain genre to justify creating a new stream and then create it.

5.2.1 AI Recommender System

There are a number of enhancements that could be added to the current AI recommender system.

5.2.1.1 Suggestion System Enhancement

The suggestion system designed for SmartRadio currently does not generate asset ratings for those users who have not rated assets. This does not have a big impact on functionality because SmartRadio performs inference rating. However, this feature could easily be added to the system. Each time the AI system detects a participating user has not rated an asset, it can generate a rating based on neighboring ratings. The formula used to generate these ratings is show in Figure 40.

$$U_x = \bar{U} + \frac{\sum_{J \in \text{Raters of } x} (J_x - \bar{J}) \cdot r_{U,}}{\sum_{J \in \text{Raters of } x} |r_{UJ}|}$$

Figure 40 - User Clustering Formula

Another enhancement for the current suggestion system would be to allow SmartRadio to suggest streams based on the current bandwidth usage. Beyond supporting multicast, SmartRadio does not make any other bandwidth reservations. It would be possible to make the suggestion system place more weight on streams that are currently active therefore possibly avoiding a situation where there are many active streams with only one listener per stream.

5.2.1.2 SmartPlaylist creation enhancements

The SmartRadio AI system currently does only genre based content filtering. This means that songs are first classified based on their genre. This feature could be improved by further classifying songs based on their respective artists or albums. This content based filtering enhancement, rooted on the premise that users have favorite artists or albums, would allow for finer grained tuning of the recommender system. SmartRadio should be able to recognize and differentiate these ratings and group them with similar artists or other albums from the same artist. For example, a user who has rated a subset of all items from set X, where set X is all assets from the same artist, would be more likely to be recommended unrated items from the same set X.

On the whole, the recommender system is based on a set of static thresholds. These thresholds are decided by the site administrator and cannot be changed without restarting the entire site. The system would benefit if these thresholds obeyed a set of rules in which they adapt to the member base. Essentially, the thresholds should be more lenient when the system is in a cold start and become more stringent as the number of assets rated increases.

5.2.2 Streamer Side

The streamer side of SmartRadio is responsible for delivering the stream to the client. Currently the streamer side support both unicast streaming and multicast streaming.

```
mIcecast Version 1.3.12 Initializing...
Starting thread engine...
[14/Sep/2002:16:13:13] Icecast Version 1.3.12 Starting..
[14/Sep/2002:16:13:13] Starting Admin Console Thread...
->    [14/Sep/2002:16:13:13]    Starting    main    connection
handler...
-> [14/Sep/2002:16:13:13] WARNING: Resolving the server name
[134.226.36.230] does not work!
-> [14/Sep/2002:16:13:13] Listening on port 8000...
-> [14/Sep/2002:16:13:13] Listening on port 8001...
-> [14/Sep/2002:16:13:13] Server limits: 900 clients, 900
clients per source, 10 sources, 5 admins
-> [14/Sep/2002:16:13:13] WWW Admin interface accessible at
http://134.226.36.230:
8000/admin
-> [14/Sep/2002:16:13:13] Starting Calender Thread...
-> [14/Sep/2002:16:13:13] Starting UDP handler thread...
-> [14/Sep/2002:16:13:13] Starting relay connector thread...
->   ->   [14/Sep/2002:16:13:13]   [Bandwidth:   0.000000MB/s]
[Sources: 0] [Clients: 0]
[Admins: 1] [Uptime: 0 seconds]
->  [14/Sep/2002:16:13:37]  Accepted  encoder  on  mountpoint
/rock from 127.0.0.1. 1
sources connected
-> START THREAD[14/Sep/2002:16:13:37] Accepted client 2 from
[127.0.0.1] on mountpoint [/rock]. 1 clients connected
->   [14/Sep/2002:16:13:38]   MULTICASTING   RTP   STREAM   ON
239.255.0.1 FROM /rock.
->    [14/Sep/2002:16:13:54]    Accepted    client    3    from
[134.226.36.223] on mountpoint [/rock]. 2 clients connected
-> [14/Sep/2002:16:14:10] Kicking client 3 [134.226.36.223]
[Client signed off] [listener], connected for 16 seconds,
239827 bytes transferred. 1 clients connected
->    [14/Sep/2002:16:19:13]    [Bandwidth:    0.033333MB/s]
[Sources: 1] [Clients: 1] [Admins: 1] [Uptime: 6 minutes]
->    [14/Sep/2002:16:21:13]    [Bandwidth:    0.033333MB/s]
[Sources: 1] [Clients: 1] [Admins: 1] [Uptime: 8 minutes]
-> [14/Sep/2002:16:22:45] Source /rock signed off, moving
clients to default mount
->  [14/Sep/2002:16:22:45]  Kicking  source  1  [127.0.0.1]
[Source signed off (killed itself)] [encoder], connected for
9 minutes and 8 seconds, 283392 bytes transfered. 0 sources
connected
-> [14/Sep/2002:16:22:45] Kicking all 1 clients for source 1
->  [14/Sep/2002:16:22:45]  Kicking  client  2  [127.0.0.1]
[Stream ended] [listener], connected for 9 minutes and 8
seconds, 8651863 bytes transfered. 0
clients connected
-> [14/Sep/2002:16:22:45] TERMINATED MULTICAST RTP STREAM ON
ï.ÿ.
```

Table 11 - Icecast console output

The streamer meets both requirements of streaming multicast streams and unicast streams.

However, it happens that while listening, the music stops streaming after an undetermined about of time. When this happens the client must restart the stream or join another stream. This bug is present both with multicast streams and unicast streams. This bug is present in the Icecast streamer and is not associated with the multicast extensions.

5.3 The Whole SmartRadio System

The SmartRadio system provides users with a scalable audio streaming solution., furthermore it is possible to setup SmartRadio as a distributed streaming solution. As illustrated in Figure 41, the clients,the website, and the streamer can all be on different autonomous system thus reducing processing use and spreading network bandwidth. Furthermore, Icecast servers can be setup to work in a hierarchical fashion, communicating between each other using unicast transmission, all servicing different autonomous systems.

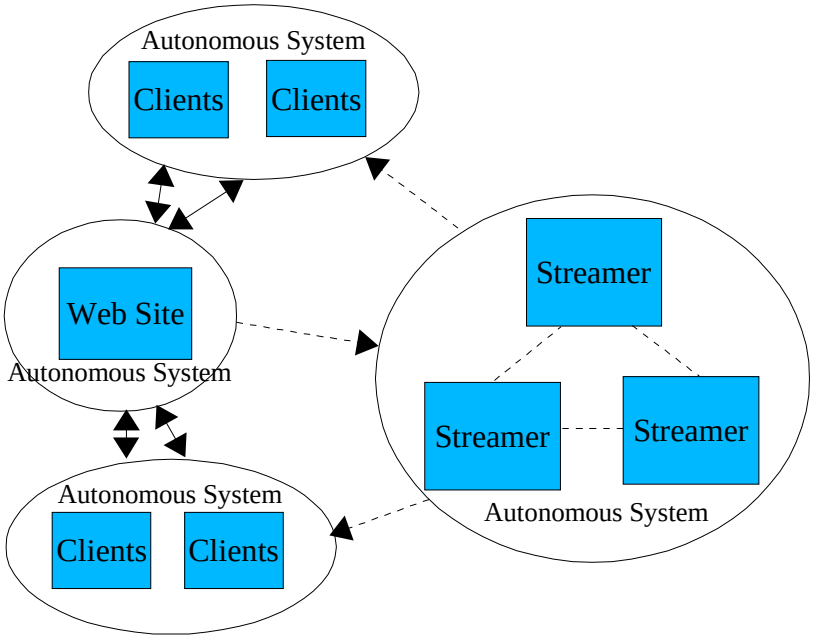


Figure 41 - Icecast Configuration

5.4 Summary

This chapter reviewed the current SmartRadio implementation. There are parts of the system that could use enhancement and others that require more work before a full release is ready.

Finally this chapter demonstrated a possible SmartRadio setup as viewed from the network level. This kind of setup maximizes the scalability features.

6 Conclusion

The pace of development is fast and furious in the world of audio streaming. Access to broadband is becoming more and more common. Computer processing power is soaring, while highly efficient, open source, audio codecs are readily available. These factors alone are causing another internet revolution. A 'multimedia' internet revolution.

Industry giants such as Cisco, Intel and many others are already combining their resources to promote Multicast. Multicast networking and its associated applications are poised to become important enablers of the multimedia revolution over both private and public networks. The power of multicast is important because it enables desired multimedia applications to scale. That is to service many users without overloading network and server resources. Multicast networking scales because it pushes network usage towards the users and away from the source.

For all the benefits of multicast services barriers to deployment exist, particularly in the public Internet. Here, both business and technical issues remain to be overcome. As a result, general Internet multicast services are likely to be a number of years away. Operators of private networks will likely add multicast only as they perform major upgrades in response to saturation of their networks with traffic. In such cases, multicast can help avoid major new expense associated with increasing the WAN's capacity. Until the Internet as a whole embraces Multicasting as a standard, streaming applications have to continue to support unicast.

Even as the scaling power of multicast networking approaches, users are burdened by another major problem. The explosion of multimedia is causing users to have to spend countless hours hunting for the appropriate content. Recommender systems are a key way to help mass customization.

Collaborative filtering combines the strengths of human intelligence in understanding information content with the speed of computers in processing that same content. Unfortunately, collaborative filtering techniques alone can be ineffective when users have not rated an item, for new users to the system, or from users who do not group well with the

opinions of others.

On the other hand Content based filtering addresses these issues by providing a common set of features extracted from the available product description. Individual customer's preferences are predicted solely from the products that he or she has rated combined with the corresponding product features. However, content based filtering can be ineffective when the items to be rated are not easily identified by a finite set of keywords.

This thesis created a scalable streaming application over the current IP multicast infrastructure and then used a hybrid collaborative and content based filtering approach for recommending and customizing dynamic multimedia content. SmartRadio uses a web based recommender system as the basis for user interaction and data collection. Streams are delivered through either Multicast or Unicast depending on the users capabilities. It is essentially a completely scalable, intelligent, audio streaming application.

7 Bibliography

[SKR] J.Ben Schafer, Joseph Konstan, John Riedl. Recommeder Systems in E-Commerce. Department of Computer Science and Engineering, University of Minnesota, 1999.

[CKLT] William K. Cheung, James T. Kwok, Martin H. Law & Kwok-Ching Tsui. Mining customer preference ratings for product recommendation using the support of vector machine and the latent class model. Department of Computer Science, Hong Kong Baptist University, Kowloon Tong Hong Kong, Intelligent Systems Reasearch Group, BT Laboratories, 1999.

[RISBG 1994] P. Resnick, N. Iacovou, M. Sushak, P.Bergstorm, and J. Riedl. GroupLen: An open architecture for collaborative filtering of netnews. In *proceedings of ACM 1994 Conference on Computer Supported Cooperative Work,* pages 175-186, Chapel Hill, NC, 1994.

[DC 1985] Deering S, Cheriton DR. Host Groups: A Multicast Extension to the Internet Protocol. RFC 966, Decemeber 1985.

[K 1999] C. Kenneth Miller. Multicast Networking and Applications. Addison Wesley Longman, Inc. 1999

[Mbone] Information on Mbone tools can be found at http://pipkin.lut.ac.uk/~ben/video/

[M 1998] D. Meyer. Administratively Scoped IP Multicast. RFC 2365, Category: Best Current Practices. University of Oregon, 1998.

[PD 2000] Larry L. Peterson, Bruce S. Davie. Computer Networks A systems Approach. Second Edition. Morgan Kaufmann Publishers. 2000.

[T 1998] Pusateri T. Distance Vector Multicast Routing Protocol Specification. Internet Draft Work in Progress, draft-ietf-idmr-dvmrp-v3-06.txt, March 1998.

[D 1989] Deering S. Host Extensions for IP Multicasting. RFC 1112, August 1989.

[MP3] Information on MPEG Audio Layer-3 can be found at http://www.iis.fhg.de/amm/techinf/layer3/index.html

[SKR] J.Ben Schafer, Joseph Konstan, John Riedl. Recommeder Systems in E-Commerce. Department of Computer Science and Engineering, University of Minnesota, 1999.

[RTP] Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, January 1996.

[ALF] Clark, Tennenhouse. Architectural Considerations for a New Generation of Protocols (context), 1990

[IPMI] Stardust technologies, Inc. 1997 IP Multicast Usage Report. IP Multicast Initiative(IPMI), July 1997.

[R 1971] Rocchio, J. Relevant feedback information retrieval. In Gerald Salton (editor). The SMART retrieval system- experiments in automated document processing (pp. 313-323). Prentice-Hall, Englewood Cliffs, NJ. 1971.

[BHK 1998] John S. Breese, David Heckerman, Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Microsoft Research. 1998

[BS 1997] Marko Balabanovic, Yoav Shoham. Content-Based, Collaborative Recommendation. Communications of the ACM, 40(3), March 1997.

[SUN] Sun Microsystems, Designing Enterprise Applications with the J2EETM Platform, Second Edition, Pearson Education Corporate Sales Division, 2002.

[JSS] Java Servlet Specification Version 2.3. Final Release, September 17, 2001. Sun Microsystems, Inc.

[RFC 2250] Hoffman, D., Fernando, G., Goyal, V. and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video", RFC 2250, January 1998.

[RFC 3119] R. Finlayson, “A More Loss-Tolerant RTP Payload Format for MP3 Audio”, RFC 3119, June 2001.

8 Appendix A: List of supported genre types

Blues, Classic Rock, Country, Dance, Disco, Funk, Grunge, Hip-Hop Jazz Metal New Age Oldies Other Pop R&B Rap Reggae Rock Techno Industrial Alternative Ska Death Metal Pranks Soundtrack Euro-Techno Ambient Trip-Hop Vocal Jazz+Funk Fusion Trance Classical Instrumental Acid House Game Sound Clip Gospel Noise AlternRock Bass Soul Punk Space Meditative Instrumental Pop Instrumental Rock Ethnic Gothic Darkwave Techno-Industrial Electronic Pop-Folk Eurodance Dream Southern Rock Comedy Cult Gangsta Top 40 Christian Rap Pop/Funk Jungle Native American Cabaret New Wave Psychadelic Rave Showtunes Trailer Lo-Fi Tribal Acid Punk Acid Jazz Polka Retro Musical Rock & Roll Hard Rock Folk Folk/Rock National Folk Swing Bebob Latin Revival Celtic Bluegrass Avantgarde Gothic Rock Progressive Rock Psychedelic Rock Symphonic Rock Slow Rock Big Band Chorus Easy Listening Acoustic Humour Speech Chanson Opera Chamber Music Sonata Symphony Booty Bass Primus Porn Groove Satire Slow Jam Club Tango Samba Folklore

9 Appendix B: List of supported media types