r-MUSIC, A Collaborative Music DJ for Ad Hoc Networks

Ursula Wolz, Michael Massimi, Eric Tarn
Department of Computer Science, The College of New Jersey, Ewing, NJ 08628
wolz@tcnj.edu, massimi2@tcnj.edu, tarn2@tcnj.edu

Abstract

We present r-MUSIC, client-server architecture for sharing music through a persistent resource in which the music data is not shared among users, but is streamed from personal digital music players to a stationary high fidelity speaker system. The client side of this system is installed on the personal digital music players. Users' music selections are transmitted through a wireless interface to the r-MUSIC server, which mediates a song title queue. Users call referenda on songs in the queue, and then vote on the popularity of songs, to mediate if and when they will be played. Our architecture includes a vote balancing mechanism that prevents users and their songs from becoming either too dominant or isolated. The power of this system is that an ad hoc group can share music without the need for a formal mediator. Mediation occurs entirely through collaboration.

1. Introduction

Portable, on-demand music is rapidly becoming ubiquitous. An emerging problem is how people can share that music spontaneously in a public social setting, akin to the classic High School "sock hop" of the 1950s. In this scenario there is no disc jockey, nor do individuals bring a premixed play list. Instead, each participant adds songs to the dynamic play list in an entirely real-time spontaneous manner. Our assumption is that the music is shared publicly (e.g. through high fidelity speakers) rather than through private individual earphones. Consequently, this is a fixed resource sharing problem akin to sharing a network printer. The novelty to our approach is in how the group decides what music to allow and in what order.

In this paper we describe an architecture called r-MUSIC, pronounced "our music" (Resource Mediation by User-Supported Initiative in Communities), that supports the creation of a dynamic and equitable security policy management system for ad hoc networks based on peer consensus of resource management. Our approach allows any group of individuals to establish a local network without a human systems administrator. From the perspective of the music application, this eliminates the need for a DJ. r-MUSIC dynamically assigns resource access rights (e.g. to the audio speakers), and through peer mediation (voting) allows the participants to determine who gets to use the resources and how often.

Our architecture supports secure music sharing. Rather than "give" a peer a song, groups only "share" the music within the spontaneous social setting. Once a piece of music is played, it continues to reside only on the electronic play list from which it originated.

2. Background and Related Work

Before describing the architecture of r-MUSIC we provide the social setting in which it would be used, articulate our assumptions, and contrast our approach with related work.

2.1. The 21st Century Sock Hop

Imagine a group of friends, all of whom have a personal digital music device. They meet at a venue where a high-quality sound system is connected to an r-MUSIC server. This server holds no music, but like a web server merely queues a play list for the *fixed resource,* namely the sound system.

Each friend brings his or her own personal device that includes r-MUSIC client software. Upon entering the room, registration software in the personal device is activated by the person so that the server will recognize the device. The server automatically assigns access privileges to each device, starting with a neutral assessment of the person's status.

Status becomes important later to maintain a socially balanced group. High status implies trust within the ad hoc network, and those with high status have more influence on the group decision-making power (e.g. whose songs are played when). Low status diminishes the users' influence on the group. This reflects the natural dynamics of an informal party in which one-person-one-vote rarely applies to the mediation of dominant, server through the interface on their personal devices. Each user's personal device displays the public list. Any member can also stage a referendum on the popularity of a song, thus asking the group to adjudicate how soon or even whether a particular piece of music should be played. They can ask to have their own song considered, or one of another member. Members can also remove songs they have posted because of peer pressure from referenda.

When a song comes to the head of the queue, it is streamed from the client's device, and funneled through the server in real time. Thus the material is never transferred to a persistent medium and maintains copyright protection.

This simple scenario can be elaborated. For example, a professional DJ could sell her services for a more formal party and provide specialized digital music players with a well-organized index of songs for particular kinds of events (teen parties, bar mitzvahs, weddings).

The immediately obvious drawback of this technology is how the queue is managed. As will be elaborated upon in section 3, a linear first in, first out, un-weighted queue cannot account for unbalanced resource demand. For example, an individual or small group with dexterous fingers can dominate the play list. Without the ability to call referenda, unacceptable music cannot be forced off the list by group consensus. In fixed networks, a human system administrator adjudicates. In ad hoc networks such as the one described here, an automatic method of creating balance in resource access is required. Section 3 describes in detail how we address this problem.

2.2 Assumptions

Our work makes a number of assumptions that should be stated at the onset. These have to do with social and communicative dynamics of small group behavior. Regardless of whether these assumptions can be proven correct, in sum, they capture the notion of listener as active rather than passive participant in a musical experience.

First, we assume that despite the awesome potential of private listening, in some social settings listeners will continue to want to share the experience of listening to a particular recorded piece. Furthermore they will want to hear the music through high-fidelity audio equipment rather than through headphones.

Second, we assume that there are many ad hoc social settings such as private parties where a group comes together to contribute music. In the 1950s and 1960s individuals brought records and stacked them. Casual submissive and aberrant behavior.
The members of the group post songs to the r-Music communication within the group determined preference as well as the ordering of the pieces. In earlier times a musicale would provide entertainment in which the participants shared their music by performing it, and the jazz club still captures the flavor of such spontaneity. In the 70s and 80s, such complete spontaneity was replaced by the advent of the "party tape", which more recently has become "burning a CD", and most recently "creating a play list." In developing the r-MUSIC technology for more mundane ad hoc network applications, we saw its potential to bring back the days of truly spontaneously shared music. Clearly partygoers will communicate face-to-face, editorializing on the proposed play list. Our technology simply affords them the requisite mechanisms for easily sharing their digital music.

Our third assumption is that there is indeed a desire for individual expression within any group. We assume that people want to take ownership of the entertainment in a social setting. Canned play lists, automatic disc jockeys and other technologies that make listeners passive do not exploit intrinsic creativity in all individuals. If people in a group can have real control of a dynamic play list, then they can express their creativity in a manner that brings satisfaction to the whole group. Granted, some will choose not to contribute to the play list. At the extreme, a participant can simply listen to what is played, but most of us are critics too, and the ability to vote on whether and when a song is played should appeal to even the most passive party participant.

2.3. Related Work

Our work extends current work on ad hoc network resource allocation [2]. Current methods of resource allocation have many drawbacks. A system administrator is needed to establish different user groups, each with distinct access rights and priority to resources, and forces all users to be categorized into an unchanging group. A queue is generally used to determine access priority to a resource. Time and the number of resources are limiting factors; they can result in less-than-optimal arrangements of allocations. There also remain ambiguities in specifying how arrangements should be made amongst parties.

Zhao and Karamcheti have previously addressed the issue of trading resource usage amongst two or more entities. They express a desire to create "sharing agreements," which act as policies. These specify the obligations, privileges, and execution constraints of each of the involved stakeholders. Further, they stress that "agreements must be enforced in the presence of heterogeneous resource types and dynamically changing user set and resource availability" [3]. Our system contributes a method for handling these dynamic user sets and resource availability; it enables this problem to be addressed on the fly by the stakeholders.

In the r-MUSIC system, users want to minimize the time spent formally establishing rules of conduct. Storey, Blair, and Friday term these situations as "active environments." They foresee a future where the promulgation of wirelessly networked devices reaches a threshold where strict systems come into play to regulate their communication. They believe that "[t]he potential explosion in numbers of devices will require careful consideration of how interaction between them is to take place, such that these devices may interact to achieve a common task" [5]. We agree with this statement, but instead of considering computerized items, we examine human behavior. We attempt to, through reflection on test results, create a robust protocol that can formalize abstract notions that "human computers" will generate in sharing situations.

Dannenberg and Hibbard discuss arbitration of resources amongst policies in a "commerce model" where all users have their own personal computers [4]. They describe a *Banker* and *Butler* who are responsible for monitoring resources and serving them to requesting parties. They briefly mention that "[f]or human engineering reasons, the user should be able to create policies that constrain sharing" of their own machine resources. What if the user does not own the machine? How are policies forged in these cases? r-MUSIC addresses this problem.

Revocation of resource rights is another issue that the Butler system is designed to address. It identifies misuse and takes one of three actions. One possible action is to warn the user that the resources are being withdrawn so that they may make preparations to leave. The second possibility is to cause a "deportation" of their process, wherein the process is transferred, uninterrupted, back to the machine that the user owns. In the r-MUSIC scenario, the process (e.g. the song) would not be deported, but instead suspended – it would resume its processing after all other requests have been filled. The final action is abortion, which is a last resort. [9]

Ahamad, Ammar, and Cheung developed the concept of "multidimensional voting" through the use of "k-dimensional vectors of nonnegative integers and each dimension is independent of the others." Their algorithm approaches the problem of quorum assignments through the use of a mathematical voting model. They prove that new voting models can be used to solve problems that traditional ones do not [1]. r-MUSIC is crafted in the same spirit.

Summers et al. [8] describe the idea of a "Resource-sharing Machine" that is an extension of the PC-DOS operating system. The concern for networked resource sharing was intact in 1985, when the article was published, as it is now. The only changes are newer software and hardware with a spontaneous network. In this article, there is also a discussion of queues, locking of resources, and other tools involved with resource sharing and allocation [6]. These methods are to be improved upon by r-MUSIC.

Damiani et al. took a reputation-based approach for choosing reliable resources in peer-to-peer networks by proposing a "self-regulating system where the P2P network is used to implement a robust reputation mechanism" [3]. The peer-to-peer management system for wearable mobile devices constructs a system of decentralized resource control where access to shared resources is not determined by a system administrator [9]. Davison and Graefe "propose a new framework for [dynamic] resource allocation based on concepts from microeconomics" where "a resource broker … realizes a profit by 'selling' resources to competing operators using a performance-based 'currency.'" [5] Our work furthers these efforts by providing a solid mathematical foundation that describes human interaction in a self-governing environment.

The benefits of such systems are significant: the overhead of a human system administrator is eliminated once the system is in place, resource reliability is increased, users are able to participate in the allocation of resources (leading to increased user satisfaction), and resources are better utilized through "profit maximization" [9].

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

Figure 1: r-MUSIC Client Interface

3. The r-MUSIC System

The r-MUSIC system is based on a server and a client architecture and their interaction. Key to the architecture is a user-status-balancing algorithm that prevents resource lockout, but more importantly, significantly diminishes the ability of any individual or sub-group to obtain such dominance as to create "scapegoats" in the group. This section describes the architecture in detail.

Section 4 reports on the status of the implementation and test results.

3.1 The Client Architecture

The r-Music client is an interface between the r-Music server and the user's digital music player that stores, organizes and selects songs. The client is responsible for registering with the server and receiving a client id upon entry into the ad hoc network. The client also explicitly "checks out" upon leaving the ad hoc network, or implicitly "checks out" when it goes 'out of range" of the server. The client receives and displays the current public play list from the server. Figure 1 shows a prototype interface that is comfortably legible on a standard handheld screen. The user switches windows to move between this screen and the normal digital music interface.

Through the r-Music client, the user can post songs and delete songs he or she previously posted. The user can also post a referendum on a song on the public play list (regardless of the source of the song).

A referendum generates a warning to the other clients in the network (e.g. a sound or flashing icon) and users have a fixed time (e.g. a minute) to vote the song "up" or "down". Voter impact on song placement in the queue is described below.

Finally, the client waits for a request from the server to begin playing a song. The song is not streamed to the server, but instead is directly streamed to the audio equipment. Hence the data for the song is never stored external to any device other than the personal digital music device of origin. There is no file copying in our architecture. When the song streaming is complete, the client informs the server so that the public play list can be updated. Consequently, once a song begins to play, only the client of origin has control of the data streaming.

3.2 The Server Architecture

The r-MUSIC server is responsible for verifying the veracity of clients, registering them, and maintaining client status. All clients enter the network with neutral status (a value of 50%). The server maintains the public play list and broadcasts the information on the list to active clients.

The server updates the play list by managing the queue, including updating song placement after a referendum, or after the current song is complete. The server only manages song ids. It never manipulates the song data file. All songs enter the queue with a neutral rating (of 50%). This rating is different from the user rating described previously. As a song progresses in the queue, its rating is increased by 1 percentage point. Hence, without any referenda, songs implicitly enter at the back of the queue.

The server also manages referenda. More than one referendum may be active at a time (users may lobby for the placement in the list of more than one song.) The server posts the referendum to all active clients, and tallies the "up" or "down" votes. The song rating is updated based on the votes, taking into account the user ratings, and the queue is revised as a result.

Finally, the server also manages the status ratings of the clients. This occurs at the close of a referendum and when a new song comes to the front of the queue.

3.3 Updating Client and Song Rating

When a user decides to request a song, the request is sent to the r-MUSIC server. Once received, the server will insert the requested song into the play list at the predetermined median ranking (50). At this point, the song appears on the public play list.

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

Figure 2: Impact of Balancing on Rating Change

Once a song appears on the public play list, any user can initiate a referendum by choosing to boost or lower this song's rating. When this occurs, notification of a new referendum is sent to all users. Users may choose to participate in a referendum or simply ignore it. If a user chooses to participate, he or she will be able to choose whether they agree or disagree with the proposed change in rating.

For instance, Mike is particularly enthused about hearing the song "Norwegian Wood." He chooses to boost its rating. When this happens, Ursula (along with every other client) receives a notification informing her that Mike wants to hear "Norwegian Wood" sooner. She is presented with two options – agree, or disagree. If she agrees, her user rating will influence the calculation of the new song's rating in a positive way. As an additional reward for choosing a song in "good taste," Mike's rating would increase a small amount as well. If she disagrees, her rating will have a negative effect on the song rating, and Mike's user rating will decrease slightly. She may choose to ignore the referendum completely and cast no vote.

At the end of a referendum, the votes determine the new rating of the song in the queue. The calculations, listed below, rely largely upon two factors – the direction of the vote (positive or negative), and the rating of the user casting this vote. This mathematically captures the idea that if someone has a high rating, they have a good sense of musical taste, and therefore will make a better decision about a particular song. Likewise, if someone has poor taste in songs (indicated by a low rating), they will have a smaller effect on how the play list can change during referenda.

With its new rating determined, the targeted song will move up or down on the play list. This ultimately means that the songs people like will be played sooner than the songs that are either neutral or disliked.

3.3 Balancing Impact

It can be shown mathematically that a linear voting system without bounds is quickly overwhelmed. This is corroborated by our results as described in section 4. We use an algorithm that changes ratings based on a number of adjustments made over time. Each of these adjustments pushes the target user or song higher or lower on the scale of credibility but in a manner that balances all ratings near a median value rather than sending some entities off the scale in either direction. We describe the formulae in terms of user ratings, but the same approach applies to song rating.

When a user joins the system for the first time, they are assigned a rating that is equal to the median value. To prevent excessively high or low rating values, we impose artificial limits on the high and low ends of the continuum. For testing purposes we chose the range 1-100 with a midpoint of 50.

The source rating is the rating value of the user wishing to change another user's rating. The target rating is the impacted user's rating before applying the change. The impact ratio was created to accommodate the fact that the relative rating of two users should come into play. This represents an extra boost, for instance, if a particularly credible user issues a lesser-ranked user a vote of confidence. The change curve was determined by taking the integral of a modified bell curve. This was necessary to avoid obtaining certain values that would result in a divide by zero. Our formulae for determining the change of a user's rating in a social resource-sharing network are:

$$T_{new} = IC$$

$$I = \frac{S}{T}$$

$$C = \frac{R^2}{|M - T| + R}$$

where T is the target rating, I is the impact, C is the change, S is the source rating, R is the maximum change permitted for a single vote, and M is the median of values. Through experimentation we have determined that R must be inversely related to the size of the group. At present we determine R by hand. Further experimentation should provide sufficient data for curve fitting and a resulting fixed formula. As can be seen from the graph, all voting tends to cluster user rank toward the median value. As rank goes to the fringe, it becomes more difficult for the group to either vote someone "up" or "down".

4. Implementation and Testing

Implementation of the complete r-MUSIC client/server software requires a combination of off-the-shelf technology and specialized code. Our focus to-date has been to develop prototype software for testing theoretical outcomes of the specialized code, especially the veracity of the balancing formula on dynamic systems. We have implemented a version of the client-server architecture that allows us to run batch jobs based on simulations of predicted user behavior. We are poised to conduct user tests on a second system that allows us to do rapid prototyping and conduct principled usability studies.

4.1 Integration with Off-the-Shelf Technology

The hardware needed to make this system a reality is readily available in the market today. We are in the process of procuring and constructing the components described here while we have implemented and tested prototypes in Java on Linux platforms. Our environment consists of two major pieces – the r-MUSIC server and r-MUSIC clients.

The server is a desktop or laptop computer equipped with IEEE 802.11b wireless access. Further, the server also includes a database system (such as MySQL) that serves as a storage area for song ids, requests, history logs, and the play list. Finally, the manager runs r-

MUSIC software that performs period checks and calculations on the contents of the database.

r-MUSIC clients are handheld devices. We have selected the Sharp Zaurus, running Linux. These clients have any number of digital music files preloaded onto their memory in a pre-determined path in the file system. Each device is additionally equipped with wireless 802.11b access as well, using a local access point. The clients then choose to run specific client software that registers the device with the manager. ID3 song information residing in the predetermined path is read into memory and passed to the server, where the song listings for each user are stored. Our Java-based user interface software is responsible for client-server communication as well as human-computer communication.

One concern we have is the rate at which streaming of music data can occur through our wireless interface. We anticipate that if the rate degrades song quality then state-of-the-art buffering can ameliorate the problem. Alternatively, entirely for testing purposes in the short term, we can store song data on the server and only store song ids on the client side, preserving our goal of containing song data entirely on a single persistent medium.

4.2 Simulating Predicted Behavior

We have implemented and tested a server solution that can execute batch jobs of user scenarios. In such scenarios, *n* users can be specified who behave in a specified manner during a "voting round." A round is defined as a time unit in which some fixed or random subset of *n,* post *s* songs, vote on *r* referenda, and another subset of fixed or random users *v+,* vote for, and *v-* vote against each referendum. Using this testing software we can show the power of our balancing formula.

In a normal system, the amount of change with each vote is a constant number, usually 1. By incorporating a formula that alters the amount of change depending on the rating of the target user, we can vary the size of the change on each vote. To compare the number of votes necessary to move a user from a midpoint (50) to an extreme (100), we calculate the area under the curve. The area under the curve for a straight line at 1 is 50. The area under the curve for a bell-shaped line from 50 to 100 is approximately 180. The straight line and bell-curve intersect at the point where the change is equal to one. It is beyond this point that the bell-curve demands more votes than the straight line in order to achieve a rating of 100, due to the diminishing size of the changes in rating.

4.3 Results of Extreme Behavior

We present results of extreme behavior because it demonstrates the upper bounds, or maximum rate at which a user can become dominant or isolated. The intent of our balancing formula is to slow the rate at which this occurs. Dominance is defined as the user reaching a rating of 100. Isolation is defined as the user reaching a rating of 1. Three cases illustrate the phenomenon of dominance and isolation with 5, 10 and 20 users. All users start off with a rating of 50 and cast one vote every round of voting for the same target user. The default step is assigned by hand relative to the number of users. For each case there are two experiments: (1) all users vote positively to push one user to dominance, and (2) all users vote negatively to push one user to isolation. Each experiment was executed three different ways:

1. Without the balancing formula (e.g. linear voting).
2. With the balancing formula applied sequentially after each vote during a round.
3. With the balancing formula applied at the end of each round based on the aggregate result of overall voting.

Table 1 shows the results of the three experiments to achieve dominance and the three experiments to achieve isolation. In all experiments the sequential application of the formula significantly reduces the impact of voting, even with 20 users. Although aggregate voting is more efficient, and intuitively appears to be fairer than sequential voting, it doesn't provide the same degree of protection from dominance or isolation that occurs with the sequential application of voting. Our results also show how isolation and dominance perform in a predictably similar fashion. It should take the same amount of time to isolate some one as to let some one dominate.

Table 1: Results of Experimentation

Number of Rounds to Achieve Dominance				
Number Of Users	Default Step	Without Formula	Formula Applied Sequential	Formula Applied as Aggregate
5	5	**2**	**12**	**5**
10	3	**2**	**16**	**5**
20	2	**2**	**17**	**3**

Number of Rounds to Achieve Isolation				
Number Of Users	Default Step	Without Formula	Formula Applied Sequential	Formula Applied as Aggregate

5	5	**2**	**12**	**5**
10	3	**2**	**16**	**5**
20	2	**2**	**17**	**3**

5. Summary and Future Work

The r-MUSIC architecture presented here has the potential to provide a rich environment in which to share music without violating copyright laws. The application of even a simple balancing formula can provide an order of magnitude improvement on preventing dominance or isolation.

Our next step with the simulation is to experiment with returning users from dominance and isolation to neutral status. A limitation of our current simulation is that we cannot adequately represent the predilections and personal preferences of human users. Preliminary experiments to model "typical users" using probabilities has been problematic. Further simulation is also required to develop a mathematical theory of the relationship between numbers of users and default step.

We have developed usability protocols and are poised to do experiments with human volunteers to test extreme behavior. For example we will target a "scapegoat" or a "charismatic leader" who some, but not all, users will vote for or against regularly, but not as systematically as our batch simulator. We also plan to conduct experiments where subgroups try to return status to isolated individuals. Finally we need to set our system loose in real situations and develop surveys and focus group protocols that identify strengths and weaknesses in our approach. We look forward to this final step, and anticipate that real users, in real social settings will find our software an exciting new entertainment medium.

6. References

[1] Ahamad, M, M. H. Ammar, and S. Y. Cheung. Multidimensional Voting. *ACM Transactions on Computer Systems*, Vol 9, No. 4, November 1991.

[2.] Cosell, B.P, P.R. Johnson, J.H. Malman, R.E. Schantz, J. Sussman, R.H. Thomas, and D.C.Walden, An Operational System for Computer Resource Sharing. *Proceedings of the fifth ACM symposium on Operating Systems principles*, Austin, TX, USA. November 19-21, 1975.

[3] Damiani, E, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. *CCS'02*, Washington, DC, USA. November 18-22, 2002.

[4] Dannenberg R. B. and P. G. Hibbard. A Butler Process for Resource Sharing on Spice Machines. *ACM Transactions on Office Information Systems*, Vol. 3, No. 3, July 1985.

[5] Davison D. and G. Graefe. Dynamic Resource Brokering for Multi-User Query Execution. *SIGMOD'95*, San Jose, CA, USA. 1995.

[6] Massimi, M. and U. Wolz. Peer-to-Peer Policy Management System for Wearable Mobile Devices. *Proceedings of the Seventh IEEE International Symposium on Wearable Computers (ISWC'03).* White Plains, NY, October 2003

[7] Storey, M. G. Blair, and A. Friday. MARE: Resource Discovery and Configuration in Ad Hoc Networks. *Mobile Networks and Applications*, 7, 2002.

[8] Summers, R. C, M. Ebrahimi, J. M. Marberg, and U. Zernik. Design and Implementation of a Resource Sharing System as an Extension to a Personal Computer Operating System. *Proceedings of the 1985 ACM SIGSMALL symposium on Small systems*, Danvers, MA, USA. 1985.

[9] Zhao T. and V. Karamcheti. Expressing and Enforcing Distributed Resource Sharing Agreements. *Proceedings of the IEEE/ACM SC2000 Conference.* Dallas, TX, USA. November 4-10, 2000.