Supporting Low-Latency Broadcast in End System Multicast

Justin Weisz
jweisz@cs.cmu.edu

Matthew Brown
mcbrown@ece.cmu.edu

Abstract

We evaluate several different overlay routing structures for supporting interactivity features in End Sysytem Multicast. Interactivity features include low-bandwidth textual chat, as well as higher-bandwidth audio conferencing and immersive audio applications. Overlays are evaluated using real interaction patterns seen in IRC, as well as real join and leave patterns from an ESM video broadcast. Results indicate that a Hybrid overlay structure, which is a combination of central server and peer-to-peer techniques, provides the smallest delay for textual messages sent through the system. Also, the Hybrid and ESM-Latency structures are best capable of supporting higher-bandwidth audio applications.

1 Introduction

End System Multicast (ESM) is a system designed to allow users to broadcast high-quality video and audio to a large number of receivers. It operates by dynamically building and refining an overlay multicast tree structure, with the source of the video situated at the root of the tree. Data from the video broadcast is sent through the tree, from root to leaves, and each non-leaf participant is responsible for maintaining a set of children, to which they forward the video and audio data.

Each participant in the tree has a certain amount of upstream bandwidth, which determines the number of children they can support. Nodes with higher amounts of bandwidth can support more children, and the tree is constructed to provide the greatest amount of bandwidth to each node, up to the bit rate of the stream. Thus, ESM optimizes the tree for bandwidth. A second optimization is made for latency, whereby when a node has to chose a new parent, lower-latency nodes are preferred over higher-latency nodes when both nodes are able to provide it the video stream [3]. It has been shown in [11] that this overlay structure has been highly successful for the transmission of high-quality video and audio streams.

However, a new generation of features is required within ESM in order to continue the success of ESM within the real world. Users need to be compelled to install the ESM player in order to view the content broadcast via ESM, and content publishers need to be convinced of the benefits of using the ESM technology before they will publish their content on ESM. In order to provide more value to both of these groups, we seek to add *interactivity features* within ESM. These features allow users to become more engaged in the broadcast, by enabling them to interact amongst themselves, as well as with the content providers. Since users feel more satisfied when able to interact with others [15], we believe that interactivity features are a necessary feature to implement within ESM.

There are two classes of interactivity features we consider. The first includes features which do not have stringent bandwidth requirements. Textual chat, buddy lists, awareness displays, shared whiteboards, and voting systems all have trivial bandwidth requirements, when compared to high-quality streaming video. The second class includes features requiring higher bandwidth, such as audio chat and immersive audio. Both of these feature classes share the same requirement: data must be delivered in a timely manner.

It has been shown in [1] that there is a maximum amount of delay tolerable in interactive communication applications (in that case, the telephone system). Thus, in order to support the two classes of interactivity features in ESM, careful consideration needs to be made in order to minimize the amount of network-imposed delay experienced by interaction messages. Because the existing ESM tree is primarily optimized for bandwidth, we propose building a separate overlay structure for the purpose of supporting low-latency transport of interaction messages. As a first step in this transport infrastructure, we support only one function: broadcasting a message to the entire set of recipients. Sending a message to a subset of hosts can be done either by client-side filtering, or by building multiple overlay structures.

It should be noted that interactivity features are not the only application of a low-latency infrastructure. A separate, low-latency infrastructure could also be used for packet recovery, adding a greater degree of resilience to the system in the presence of link failures. Configuration changes, such as changing the bit rate of the video, could also be broadcasted using this infrastructure as well, giving hosts some time to reconfigure themselves before the changes take effect.

Our project aims to evaluate several different overlay routing structures for their ability to deliver interactivity messages in a timely manner. Specifically, we compare a centralized server architecture with a fully connected mesh, a multicast tree based on the existing ESM bandwidth optimizations, a multicast tree optimized for latency, and a hybrid scheme proposed in [2] which mixes central server and peer-to-peer techniques. To evaluate each of these overlay routing structures, we first study user interactions on Internet Relay Chat

(IRC) [13], a popular text chat system. Probability models are developed for the distribution of message interarrival times and message sizes, and these models are used to generate workloads representative of the behavior patterns seen in IRC. We next implement the overlay routing structures in a simulator, and evaluate their performance using data from an ESM broadcast.

The rest of this paper is organized as follows. Section 2 discusses related work. The IRC study, probability models and generated workloads are discussed in Section 3. A description of the different overlay routing structures, the data used from ESM, and the simulator is presented in Section 4. Our experiments and their results are presented in Section 5, and we conclude in Section 6.

2 Related work

End System Multicast lies at the foundation of our work, and it has been published in [3, 4, 11].

Our approach of evaluating different overlay routing structures in terms of their performance is motivated by [9], in which the authors perform an analysis of different DHT routing algorithms.

In [2], the authors present the "hybrid" routing algorithm for immserive audio events, where nodes ship their audio data to the central server, but also send it in a unicast fashion to those nodes who are close-by in the network. We build off of their work by examining just how well the "hybrid" scheme performs when global knowledge of the underlying network latencies are known.

The authors of [14] propose an algorithm for building a low-delay application multicast tree, by rearranging nodes within a local region into different configurations, and then scoring those configurations using a function which minimizes the maximum delay from the root node to each node in that region. Their evaluation of this algorithm is based on link stress over different network topologies, rather than application-level performance. We leave an implementation of this algorithm open as future work.

Perhaps one of the earliest studies of the statistical behiavor of users was performed in [5], where they find that the interarrival times for processes on a time-sharing system follows an exponential distribution. It is interesting to contrast how humans interact with a computer, where commands arrive following an exponential distributionj, versus how humans interact amongst themselves, which follows a heavy-tailed distribution.

Another study which looks at the statistical behaviors of users focuses on multiplayer games. In [10], the authors find that player duration times follow an exponential distribution, but their interarrival times fit a heavy-tailed distribution.

The authors of [7] perform a study of Internet chat systems, and their primary contribution is a method for separating chat traffic from other Internet traffic on an Internet link. However, they also conduct a brief analysis of message interarrival times and packet sizes, and their results strongly agree with our findings in Section 3. While they simply conclude that message interarrival times follow a heavy-tailed distribution, their data collection procedure includes chat traffic from applications other than IRC. This allows us more freedom when generalizing our results.

3 Interactivity models

In order to test the overlay structures with a workload characteristic of the patterns of interaction we expect to see during an ESM broadcast, we perform a study of how users interact on Internet Relay Chat (IRC). Data from IRC is used to create a probability distribution of message interarrival times, as well as of message sizes. To study the effects of higher-bandwidth applications, we also transform IRC messages into audio messages, and create a separate probability distribution for these.

3.1 Methodology

Data from IRC were gathered from 16 IRC channels during a 24 hour period from April 7th to April 8th, 2004. The XChat IRC client [17] was modified to record timestamps in milliseconds[1], in order to observe any effects occuring at the sub-second level. Logs were collected from a variety of IRC servers and channels, including general chat, sci-fi, video games and Internet radio[2]. Table 1 presents a summary of the IRC servers and channels used for logging, and Table 2 gives a detailed account of the amount of data collected from each channel, as well as the average message interarrival times and average message sizes seen in each channel.

3.2 Interarrival time distribution

In order to understand the nature of the interactions which take place on IRC, we analyze the probability distribution of message *interarrival times.* An interarrival time is computed as the difference in timestamps of a message and the message preceeding it, for a single user. We use the symbol I to represent a random variable for an interarrival time. Correlations between message arrivals among multiple users are not taken into account[3].

Figure 1 shows the complementary cumulative distribution function (CCDF) for interarrival times over all of the IRC

[1]Timestamps are actually recorded as a tuple of <seconds, microseconds>, which are then converted to milliseconds.

[2]Anecdotally, these topics were chosen because they seemed to be the most popular when performing a `/list` operation on each IRC server, as opposed to channels with 1500 people sharing files and not talking.

[3]This is a subject for future work.

Server	Channels	Description
broadway.ny.us.dal.net	#chat-world, #usa #cybersex #teen	General chat Adult chat Teen chat
events.scifi.com	#farscape, #madhouse, #startrek	Sci-fi
Ky1.EnterTheGame.Com	#etg, #ggl, #ut, #wow	Video game
london.uk.eu.irc.inteliinternet.com	#diradio, #hardcore	Internet radio
vortex.slashnet.org	#pr0k, #slashdot	General chat

Table 1: **Summary of IRC servers and channels**. Topics were selected to provide a representative sample of different IRC user demographics.

Channel	Logging duration	Number of interarrivals	Avg. intermessage delay, minutes (stddev)	Number of messages	Avg. message size, bytes (stddev)
#chat-world	14h 1m 2s	5541	4.4 (25.3)	6304	27.3 (23.1)
#cybersex	13h 59m 15s	2220	5.9 (27.6)	2569	33.3 (37.1)
#teen	13h 57m 37s	3588	4.4 (24.6)	3958	33.5 (24.3)
#usa	3h 0m 21s	1322	1.6 (4.2)	1446	29.3 (20.0)
#farscape	21h 7m 3s	3506	4.8 (49.6)	3592	27.0 (26.7)
#madhouse	21h 6m 50s	3750	3.5 (44.1)	3817	21.0 (24.8)
#startrek	21h 5m 11s	1282	5.5 (53.4)	1302	30.6 (38.0)
#etg	21h 9m 58s	5217	10.5 (69.1)	5663	28.1 (30.0)
#ggl	21h 9m 2s	1010	15.1 (88.4)	1108	25.7 (18.8)
#ut	21h 10m 0s	12672	5.3 (48.2)	13370	25.8 (23.8)
#wow	21h 9m 38s	3349	4.0 (42.4)	3446	22.9 (23.9)
#diradio	20h 44m 44s	3193	7.6 (60.0)	3299	27.0 (27.3)
#hardcore	20h 45m 22s	1685	7.4 (63.6)	1732	20.3 (24.2)
#talk	20h 32m 10s	31	53.7 (134.3)	44	26.4 (22.3)
#pr0k	8h 37m 59s	901	4.0 (14.5)	934	29.9 (28.2)
#slashdot	20h 46m 43s	1938	6.7 (63.4)	1990	29.7 (32.4)
Total:	284h 22m 55s	51205	5.8 (48.9)	54574	27.0 (26.3)

Table 2: **Summary of IRC logs**. This table presents the duration of logging, the number of interarrival events, the average message interarrival time, the number of messages, and the average message size for each channel. Interarrivals are computed by taking the difference of the timestamp from the current message sent by a user and the last message seen by that same user. Note that the number of interarrivals does not match the number of messages seen in each channel. This effect is due to users who send only one message.

data, on a log-log scale. The CCDF is a graph of t vs. $Pr(I \geq t)$ (i.e. the probability that some interarrival time will be greater or equal to the value on the x-axis). When computing this probability distribution, we only consider interarrival times for which there are 100 or more observances. This removes outliers from the data. It should be noted that the CCDFs for each of the IRC logs have the same basic shape as Figure 1. Logs containing more data have a more well-defined shape; thus, Figure 1 is the clearest picture of the interarrival distribution.

The most notable feature of this distribution is that it is very heavy tailed, as shown by the straight line starting at around 10,000ms. Also, the plateau between 1ms and 1000ms indicates a fairly uniform distribution (i.e. $Pr(I \geq t) \simeq Pr(I \geq t + 1)$). This function consists of three underlying probability distributions, which can be exposed by conditioning over ranges of time. Conditioning asks the question "given that I falls in the range $[min, max]$, what is the probability $Pr(I \geq t)$ for $min \leq t \leq max$". These ranges, as well as the method used to determine their boundaries[4], are listed in Table 3. It should be noted that we start our analysis at 2 milliseconds in order to filter out messages sent with interarrival times of 0ms and 1ms. These messages contained duplicate content, and we conjecture that these messages were the result of a script running on a user's machine.

Figure 2 shows the conditional interarrival time probablity

[4]This method is performed by hand, which may not be as accurate as other methods. However, the point is to show that this function is really a combination of three different underlying probablity distributions which result from conditioning on a range of time.

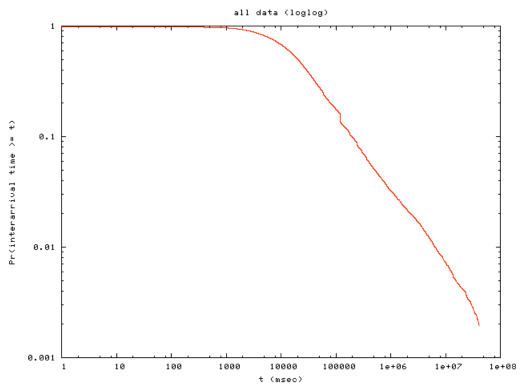

Figure 1: **CCDF of interarrival times.** This graph shows t vs. $Pr(I \geq t)$ on a log-log scale, where t is time in milliseconds, and I is a random variable for an interarrival time. The straight segment starting at 13758ms represents a heavy tail.

Range (ms)	Distribution	# in range	$Pr(I \in \text{range})$
2-1001	Quadratic	1939	0.04
1002-13758	Quadratic	18433	0.36
13759-41142219	Pareto	30833	0.60
Total:		51205	1.00

Table 3: **Interarrival time distribution ranges**. Range boundaries are determined by computing the derivative of the curve at each point, and finding the point where the slope changes to match the picture. For example, between the 1-1001ms range and the 1001-13758ms range, the slope changes sharply, and there is a large negative derivative. But, the slope becomes fairly constant in the 13758-41142219ms range, and the derivative changes only slightly in this range. The probabilities that I lies in a particular range, $Pr(I \in \text{range})$, are determined by the number of interarrivals observed in that range.

distributions. These result from assuming that I falls within each range, and then plotting the resulting probability distribution. These distributions are interesting because they show how a person's interactions follow a fundamentally different probability distribution depending on how engaged they are in the conversation.

3.3 Message size distribution

To be able to generate message sizes following the same distribution as the text messages seen in IRC, we perform the same analysis on the message size data. We use the symbol S to represent a random variable for a message's size. Figure 3 shows the CCDF for message sizes over all of the IRC data, and Table 4 presents summary statistics. We do not remove outliers from this data set because their number is

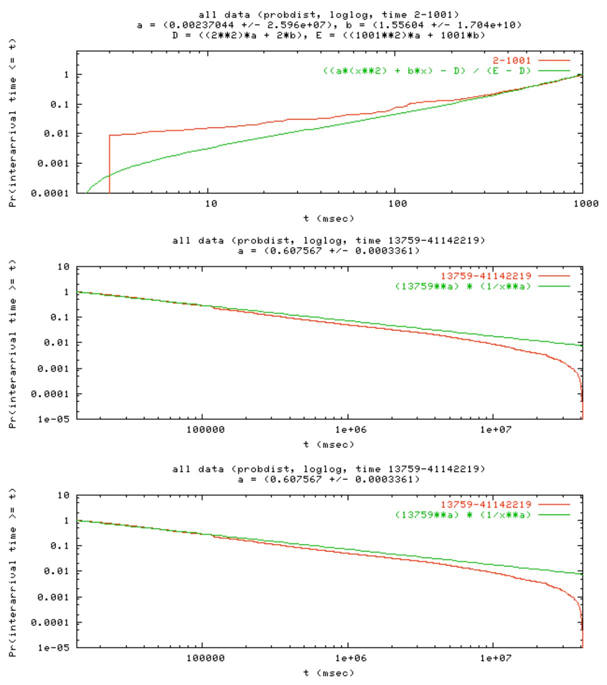

Figure 2: **Conditional distributions of interarrival times.** All graphs are on a log-log scale. Note that the ranges 2-1001ms and 1002-13758ms are presented as CDFs, while 13769-41142219ms is presented as a CCDF for clarity. The curve fit lines are used for generating data based on these distributions. This process is explained in more detail in Section 3.5.

insignificant.

Message sizes	
Count	54574
Mean	27.03 bytes
Std. dev.	26.26 bytes
Min value	0 bytes
Max value	456 bytes

Table 4: **Message size summary statistics.**

As with the interarrival data, message sizes also follow a heavy-tailed distribution. This function consists of two underlying probability distributions, which are split among the ranges of 1-35 bytes and 36-212 bytes. Table 5 lists these ranges along with the underlying probability distributions.

Figure 4 shows the conditional message size probablity distributions. As with the conditional interarrival distributions, these distributions result from assuming that S falls within each range, and then plotting the resulting probability distribution. This distribution is interesting because it shows that there are a significant number of messages less than 36 bytes, which is about four or five words (in English).

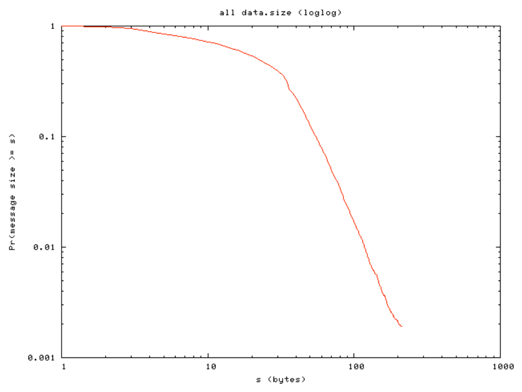

Figure 3: **CCDF of message sizes.** This graph shows s vs. $Pr(S \geq s)$ on a log-log scale, where s is the message size in bytes, and S is a random variable for the message size.

Range (bytes)	Distribution	# in range	$Pr(S \in \text{range})$
1-35	Pareto	39886	0.73
36-456	Pareto	14688	0.26
Total:		54574	1.00

Table 5: **Message size distribution ranges**. Range boundaries are determined using the same method as the interarrival time conditional distributions.

3.4 Audio workloads

As shown in Table 2, the average size of a text message is around 27 bytes. This does not pose a significant challange for any networked system, as evidenced by an IRC server's capability to handle thousands of simultaneous users using only 1-2 Mbps of bandwidth [6]. Given that the average interarrival rate of messages is almost 6 minutes, it seems unnecessary to set up and maintain any kind of overlay routing structure to service these small, infrequent text messages. However, one of our goals is to support higher-bandwidth interactivity applications such as audio chat and immersive audio, and these applications present the system with a significantly greater amount of load than just text messages. Thus, we analyze the performance of our system on workloads designed to represent these higher-bandwidth interactions.

It should be noted that our method is not fully generalizable to how people truly interact. Textual chat conversations do not share many of the characteristics which define audio conversations. For example, in face to face interactions, there are contextual clues used by participants to facilitate turn taking; one consequence of this is that one person generally speaks at one time [16]. Sometimes there is a degree of overlap among speakers, but it is hardly ever the case where a room full of people will all be speaking simultaneously.

However, we believe this caveat is minor when considering

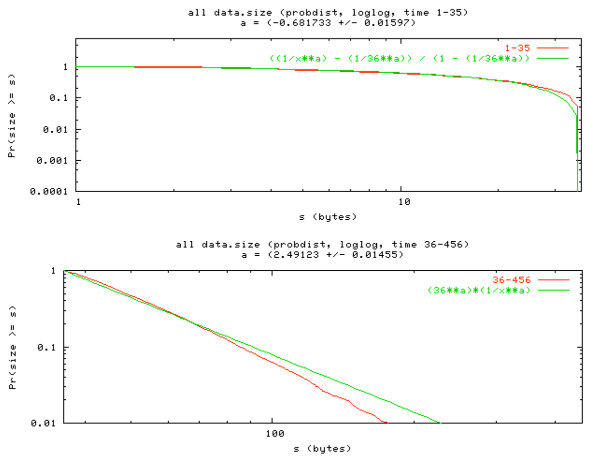

Figure 4: **Conditional distributions of message sizes.** All graphs are on a log-log scale. The curve fit lines are used for generating data based on these distributions. This process is explained in more detail in Section 3.5.

an immersive audio application. In this type of application, a user is placed in a virtual world, and surrounded by audio events which occur in this world. An immersive chat application could be created as follows: users wishing to converse move to the same location in the virtual chat room, and thus are within each others' hearing ranges. However, audio from nearby conversations is also mixed in to form an audio scene for each user, thus giving the user a feeling of presence in the virtual space, and a sense of connnection to the other users in the room. In order to create this audio scene for each user, all[5] audio needs to be delivered to all users in the virtual chat room. Thus, a broadcast infrastructure providing low-latency guarantees is required to support this type of application.

In order to generate data representing audio workloads, we use the following procedure. Text messages from IRC are converted into an audio file using a text-to-speech application[6], such that the text message is now an audio file with a computer voice speaking the message. This provides an approximation for how long a person would take to speak a message. Next, the size of the resulting audio file is used as the message size.

Performing this procedure over all of the IRC data yields the distribution shown in Figure 5. To remove outliers from our data, we consider only those audio message sizes for which there are 100 or more observances. Summary statistics of this distribution are presented in Table 6. Interestingly, this distribution looks very similar to the interarrival time distributions, although further analysis is needed to determine if a correlation exists between the interarrival time of a message

[5]Up to the point at which sound waves degenerate to the point of inaudability.

[6]We use Apple's Text to Speech API to convert text messages to audio.

Audio message sizes	
Count	46230
Mean	96982.9 bytes
Standard deviation	109203.4 bytes
Min value	502 bytes
Max value (sans outliers)	924334 bytes
Max value	2639046 bytes

Table 6: **Audio message summary statistics.** Note that the number of messages in this category is significantly lower than the count in Table 4. The reason for this discrepancy is because some IRC messages are inherantly not speakable, such as the puncuation characters from which smilies are composed. However, we interpret such messages as conveying a facial expression or gesture which would normally not be spoken in a face to face conversation. Thus, we do not include these messages in our analysis.

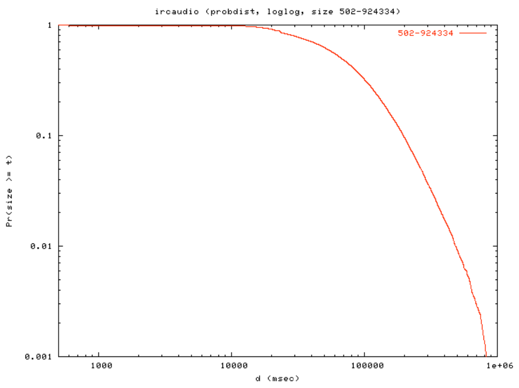

Figure 5: **CCDF of audio message sizes.** This graph shows s vs. $Pr(S \geq s)$ on a log-log scale, where s is the audio message size in bytes, and S is a random variable for the message size.

and the size of a message. Table 7 shows the ranges of the conditional probability distributions, as well as the distribution types. Figure 6 shows the conditional audio message size probablity distributions.

3.5 Workload generation

Using the models discussed in the previous section, we now generate the interactivity workloads for the overlay routing structures. Each distribution is generated using the inversion method [12], whereby a uniform random variable $u \in U(0,1)$ is transformed into a random variable drawn from our distribution.

The specific parameters used for generating workloads from each distribution are obtained by fitting a curve[7] to the CCDF, and using the resulting curve-fit parameters. Figures 2, 4 and 6 show the results of fitting a curve to these distributions. A summary of the various parameters used by each distribution is presented in Table 8.

[7]All curve fits are done with `gnuplot` [8].

Range (bytes)	Distribution	# in range	$Pr(S \in \text{range})$
502-12598	Pareto	1118	0.02
12599-100000	Pareto	30158	0.66
100001-924334	Pareto	14954	0.32
Total:		46230	1.00

Table 7: **Audio message size distribution ranges**. Range boundaries are determined using the same method as the interarrival time conditional distributions.

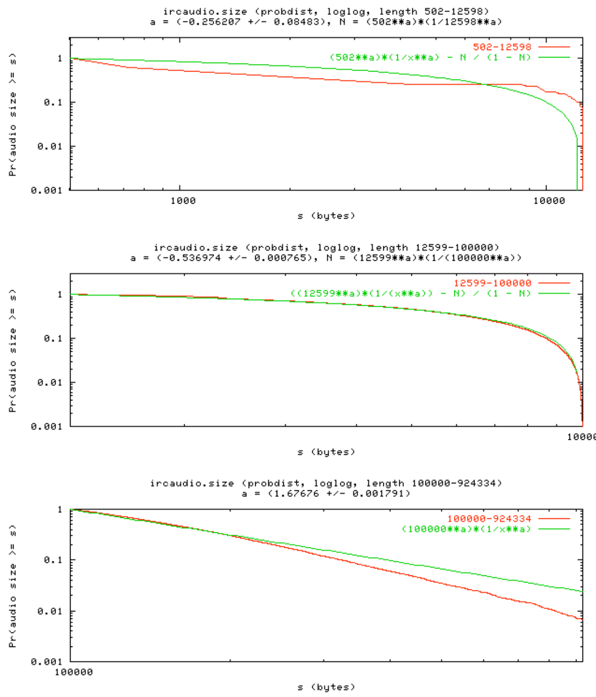

Figure 6: **Conditional distributions of audio message sizes.** All graphs are on a log-log scale. The curve fit lines are used for generating data based on these distributions. This process is explained in more detail in Section 3.5.

We now briefly describe how random variables are generated for the Bounded Pareto and Quadratic distributions, and then present the parameters for our distributions of interarrival times, message sizes, and audio message sizes. We denote our distribution functions as `combined_3` for interarrival times, `combined` for message sizes, and `ircaudio` for audio message sizes. In Section 5, we compare our interarrival and message size distributions to `exponential` and `uniform` distributions[8].

[8]These distributions are generated to have the same mean as the combined distributions.

3.5.1 Generating a Bounded Pareto distribution

A random variable X drawn from the Bounded Pareto distribution has the CCDF $F^{-1}(x) = \frac{j^{\alpha} \cdot \frac{1}{x^{\alpha}} - N}{1-N}$, where $N = \left(\frac{j}{k}\right)^{\alpha}$ (a normalization parameter), j is the initial value of the distribution, k is the upper bound, and α determines the shape of the curve. To generate a value x from the Pareto distribution, let $u \in U(0,1)$ be drawn at random, and $M = \left(\frac{k}{j}\right)^{\alpha}$. Then, $x = \left(\frac{k^{a}}{M+(u \cdot (1-M))}\right)^{1/\alpha}$. In case an upper bound is not needed, the N parameter is dropped from the CCDF, and $x = \frac{k}{u \cdot 1/\alpha}$.

3.5.2 Generating a Quadratic distribution

A random variable Y drawn from the Quadratic distribution has the CCDF $F^{-1}(x) = \frac{ax^2+bx-D}{E-D}$, where $D = aj^2 + bj$, $E = ak^2 + bk$, j and k are the lower and upper bounds on the distribution, and a and b determine the shape of the curve. To generate a value x from the Quadratic distribution, let $u \in U(0,1)$ be drawn at random. Then, x is the positive solution to $\frac{-b \pm \sqrt{b^2+4au(E-D)+4aD}}{2a}$.

3.5.3 IRC workloads

Table 8 presents the parameters used for generating workloads modeled after the `combined_3` intearrival time distribution, the `combined` message size distribution, and the `ircaudio` audio message size distribution.

4 Overlay evaluation

4.1 Overlay Structures

We consider each of the following overlay routing structures in our analysis.

Central Server. In a central server model, nodes send their messages to the central server, and the central server forwards those messages to all other nodes. The latency incurred by a message in this scheme is the delay between the source and the central server, and the central server to each destination.

Fully Connected. In a fully connected overlay, nodes simply send messages directly to every other host. The latency incurred by a message in this scheme is simply the delay between the source and each destination.

ESM-Bandwidth. The existing ESM tree can be used to send messages to all participants in a broadcast, even though this tree is optimized for bandwidth. To send a message using this scheme, a node simply sends it to the source node, and then the source node sends the message along the existing broadcast tree, using the same paths as for the video stream.

ESM-Latency. Instead of optimizing for bandwidth, we can create a separate overlay structure which optimizes for latency. This scheme behaves similar to the Narada protocol [4] in that it builds a shortest-path spanning tree rooted at each node in order to determine the lowest-cost (lowest-latency) paths to all nodes.

Hybrid. A "hybrid" delivery architecture is proposed in [2]. In the hybrid model, when a node wants to broadcast a message, he sends the message to a central server, as well as to a set of "nearby" hosts. In [2], geographic distance is used as the distance metric, where "nearby" is defined as located in the same country. When a message is sent to a central server in some other country, that message does not need to return to the country of origin, as the originator of the message can simply forward it directly to those nodes in the same country. In our case, we use latency as our distance metric. When a node sends a message, he will forward a message to the central server, as well as to a set of nodes with latencies better than the delay between the himself and the central server, plus the delay from the central server to the destination node. This ends up looking like the fully connected overlay for close neighbors, and a central server for distant neighbors.

4.2 ESM Data

Our simulations are all based on measurements taken during the ESM Slashdot event in 2002 [11]. We use four sources of data:

makeTree.child. This file contains the ESM tree over time during the Slashdot event.

JoinLeaveStayTime. This file contains the join and leave times for every node. This was used as the master file to determine which nodes were actually in the tree.

n2delay. This gives a partial listing of latencies measured between nodes in the ESM tree. Because this is a partial mapping, we filled in the missing data in this file to ensure connectivity, given any subset of nodes. Our procedure for filling in the missing data is outlined in Section 4.3.

Upstream Bandwidth Estimates. This data provides information about the number of children each node is capable of supporting during the event. It is used to provide a rough estimate of the available bandwidth at each node.

4.3 n2delay

When analyzing the Slashdot data, we discovered that the `n2delay` file contained around $(n^2)/10$ entries, some of which were zero. Because we require a complete set of n^2 latency measurements for our simulation, we fill in the missing values in the following manner:

```
$hardness \leftarrow initialhardness$
**while** nodes have no latency **do**
  **if** not making progress **then**
    reduce hardness
  **end if**
  **for all** $i, j | N(i,j) notset$ **do**
```

Distribution	Range	Type	Parameters
Interarrival	2-1001	Quadratic	$a = 0.002, b = 1.556$
(“combined_3”)	1002-13758	Quadratic	$a = 0.089, b = -4200$
	13759-41142219	Pareto	$\alpha = 0.607$
Message size	1-35	Bounded Pareto	$\alpha = -0.682$
(“combined”)	36-456	Pareto	$\alpha = 2.491$
Audio message size	502-12598	Bounded Pareto	$\alpha = -0.256$
(“ircaudio”)	12599-100000	Bounded Pareto	$\alpha = -0.537$
	100000-924334	Pareto	$\alpha = 1.677$

Table 8: **Parameters used for generating interactivity workloads**. These parameters, as well as the ranges when generating bounded distributions, were used with the inversion methods discussed in Section 3.5 to generate the interactivity workloads.

```
    if there are more than hardness valid paths from i to j through some other node k then
      $N(i, j) = mean(allsuchpaths)$
    end if
  end for
end while
```

This algorithm iteratively attempts to fill a given latency measure from $A \rightarrow B$ by averaging all paths $A{\rightarrow}K + K{\rightarrow}B$. Because there initially may be very few such paths, we begin by allowing values to only be set if there are more than a given number of valid paths to average. We call this number the hardness. It is necessary to reduce the hardness over time to ensure the convergence of this algorithm.

4.4 Simulator

To evaluate each overlay for its performance on each of the interactivity workloads, we have written an event based simulator. The simulator takes as input the processed `n2delay` file, the bandwidth file, the join leave pattern, and the interactivity workload. Whenever a node sends a message, the overlay broadcasts the message in the manner described in Section 4.1, and records the latency required to reach each node. In addition, we keep track of the bandwidth requirements of each message as it propagates through the overlay. Because the primary focus of this research is to determine which topologies are capable of achieving the least average latency, we do not prevent nodes from exceeding their available bandwidth. We keep track of the bandwidth used over time and how it compares to available bandwidth over time. We list this as a separate metric to highlight the advantages of using more intelligent overlays such as hybrid and ESM-latency.

5 Experiments

In order to determine the effects of interarrival distribution and message sizes on the overlay network, we perform the following experiments:

Experiment 1. *Evaluate a single overlay using multiple interarrival distributions, and multiple message size distribution.*

Here we evaluate the ESM-Latency overlay routing scheme using the `combined_3` and `exponential` interarrival distributions, and the `combined` and `uniform` message size distributions. Figure 7 shows that there is effectively no difference in message latencies given different distributions of message sizes and interarrival times. We believe that this may be due to a lack of fidelity in the simulator. Essentially, because all messages are delievered in exactly the same manner, independent of their size, changes in message sizes will not affect the latencies experienced by those messages. We also note that there is effectively no change in latency due to differences in interarrival distributions. We believe this is also due to a lack of fidelity in bandwidth calculations. Changes in interarrival time distributions may result in periods of a larger number of messages being sent close in time. These periods of high activity increase required bandwidth, and should affect latency due to queuing and other affects at the ESM layer.

The same is true for interarrival time. The primary affect changes in interarrival distribution should have is to produce situations of many messages arriving in close succession, causing congestion.

Experiment 2. *Evaluate multiple overlays using the same interarrival distribution and message size distribution.*

In this case, we compare all five overlay schemes using the `combined_3` interarrival distribution and the `combined` message size distribution. Figure 8 shows the Hybrid overlay scheme outperforming all other schemes. The Hybrid scheme achieves a 50% delivery rate in roughly 60ms, and a 90% delivery rate in just over 100ms. This is most likely because the Hybrid structure has only a 2 level hierarchy, compared to the ESM-Bandwidth and ESM-Latency structures, which both have a multi-level hierarchy. The Hybrid scheme performs better than the Fully Connected and Central Server schemes because it essentially chooses for each host which of those two schemes would perform better.

Experiment 3. *Evaluate the `ircaudio` workload using each overlay with both `combined_3` and `exponential` interarrival time distributions.*

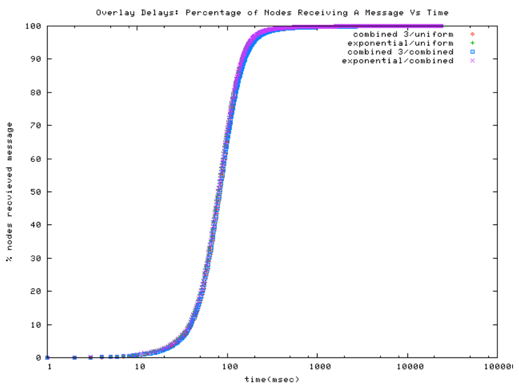

Figure 7: **Effect of Workload Distribution on Latency.** This graph shows the percentage of people who have received a message over time. It is shown on a log scale. Note that the different interarrival distributions do not have any effect on the latencies experienced by the messages.

The previous evaluations focused solely on how the topology affects latency, ignoring bandwidth usage considerations. This final experiment is aimed at providing some indication of how often each overlay exceeds the available bandwidth in the system, by using the audio message workload described in Section 3.4. Figure 9 shows each of the ten overlay/distribution combinations.

The ESM-Bandwidth scheme performs the worst, and we believe this results from having every node send their messages to the source of the tree, which then becomes overloaded. Additionally, the ESM overlay is essentially a playback of the Slashdot event; while our join/leave patterns match those of the event, our bandwidth and latency measurements are static. This may have caused problems, as the ESM overlay restructures itself based on a changing set of measurements.

The ESM-Latency performs very well in this experiment because its fanout is partially determined by the amount of available bandwidth. Finally, there appears to be a bug the code that generates the bandwidth usage data, as a result, we do not believe the graphs for Fully Connected are accurate.

6 Conclusion

In order to support a future generation of *interactive* applications built on top of the ESM framework, we have evaluated several different overlay routing structures for supporting low-latency broadcast. This evaluation was based on actual, observed data gathered from Internet Relay Chat (IRC), and several probability models were developed to reflect the distribution of message interarrival times and message sizes, as well as a distribution for the amount of audio data which would need to be sent in interactive audio applications.

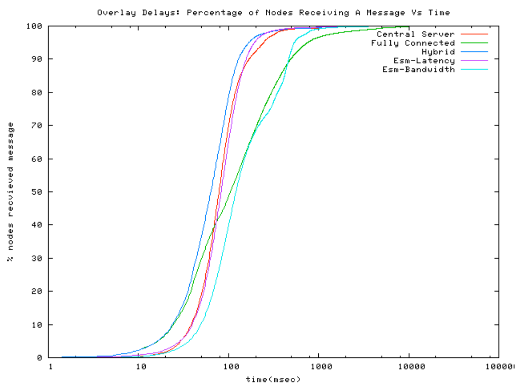

Figure 8: **Latency Comparison of Overlays** This graph shows the percentage of people who have received a message over time. It is shown on a log scale. The Hybrid scheme outperforms all other schemes, achieving a 90% message delivery rate in just over 100ms.

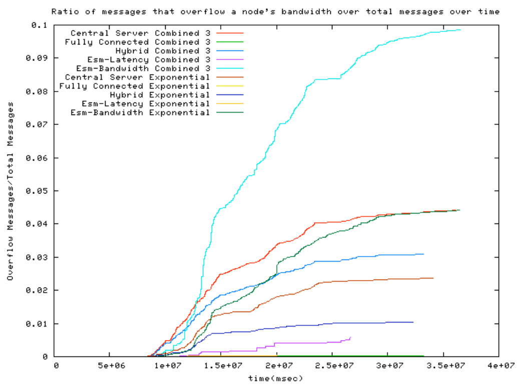

Figure 9: **Cumulative Percentage of Message Exceeding Bandwidth** This graph shows the cumulative percentage of messages that require more bandwidth than is available at a node.

From these probabilty models, we generated interactivity workloads, representing the kinds of interactions we would expect to see during an ESM broadcast. Coupling these workloads with actual ESM join/leave and network measurement data allowed us to simulate each overlay structure and evaluate it's performance in terms of latency, and in terms of how well it works for interactive audio applications.

We have discovered that the Hybrid model provided the smallest amount of delay when broadcasting small chat messages. Also, we have confirmed that naive overlays, such as Central Server and ESM-Bandwidth quickly become overloaded in the presence of higher-bandwidth audio applications. Smarter overlays, such as the Hybrid model and ESM-Latency are able to better cope with the higher demand.

References

[1] A. Aldini, M. Bernardo, R. Gorrieri, and M. Roccetti. Comparing the qos of internet audio mechanisms via formal methods. *ACM Trans. Model. Comput. Simul.*, 11(1):1–42, 2001.

[2] P. Boustead and F. Safaei. Comparison of delivery architectures for immersive audio in crowded networked games. To appear in NOSSDAV 2004.

[3] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–67. ACM Press, 2001.

[4] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clara, CA, June 2000. ACM.

[5] E. G. Coffman, Jr. and R. C. Wood. Interarrival statistics for time sharing systems. *Commun. ACM*, 9(7):500–503, 1966.

[6] The DALnet IRC Network Server Application. `http://www.dal.net/admin/vote/guidelines.php3`.

[7] C. Dewes, A. Wichmann, and A. Feldmann. An analysis of internet chat systems. In *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, pages 51–64. ACM Press, 2003.

[8] gnuplot homepage. `http://www.gnuplot.info`.

[9] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 381–394. ACM Press, 2003.

[10] T. Henderson and S. Bhatti. Modelling user behaviour in networked games. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 212–220. ACM Press, 2001.

[11] Y. hua Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an internet broadcast system based on overlay multicast. Technical report cmu-cs-03-214, December 2003.

[12] The Inversion Method. `http://www.jesus.ox.ac.uk/~clifford/a5/chap1/node5.html`.

[13] IRC.org. `http://www.irc.org`.

[14] S.-W. Tan and G. Waters. Building low delay application layer multicast trees. In M. Merabti and R. Pereira, editors, *Proceeding of 4th Annual PostGraduate Symposium: The Convergence of Telecommunications, Networking & Broadcasting*, pages 27–32. EPSRC, Liverpool John Moore University, June 2003.

[15] H.-H. Teo, L.-B. Oh, C. Liu, and K.-K. Wei. An empirical study of the effects of interactivity on web user attitude. *Int. J. Hum.-Comput. Stud.*, 58(3):281–305, 2003.

[16] B. Whittaker, S. & O'Conaill. *The role of vision in face-to-face and mediated communication.* Lawrence Erlbaum Associates, 1997.

[17] XChat IRC Client. `http://www.xchat.org/`.