CS 529 Spring 2015

Receiver-Driven Layered Multicast Project

Chaima Jemmali
cjemmali@wpi.edu

ABSTRACT

In this study, we are interested in Receiver-Driven Layered Multicast protocol and its architecture. We build an application that simulates the protocol which consists of the server sending different qualities of a video on different channels and the client will subscribe to a number of channel depending on his available bandwidth. Then we run tests and analyze data to determine the optimal waiting time for the client to drop or to add a channel.

INTRODUCTION

While most of the IP communication on the internet is between one sender and one receiver, some applications, especially real-time multimedia need to send data to multiple receivers. The problem is that internet networks are not homogeneous and bandwidth can drastically vary from one end to another. This problem makes multicasting a uniform signal to all the receivers inefficient. In fact, this will create congestion in low-capacity regions and underperformance for high-capacity regions.

In this approach, the sender broadcasts at some fixed rate without regard to changing network conditions. A better approach would be to adjust the transmission rate to match the available capacity in the network, and react to network congestion. But in the case of multicast, the notion of network capacity is ill defined, as the source cannot adapt to conflicting requirements from heterogeneous receivers.

An alternative approach is a network protocol called Receiver-driven Layered Multicast or RLM. For video, this means layering the video signal such that a base layer provides a very coarse picture quality and subsequent layers can be added, with each picture enhancing the video quality. Receiver-driven means that the client will adapt the amount of data it receives considering his available bandwidth. The sender sends multiple copies of the same video simultaneously at different rates (resulting in different qualities), each receiver then runs a simple control loop; on a spare network capacity add a layer, upon congestion drop a layer. Now the question is how the client determines if there is congestion and how long does he need to wait to assume that the network capacity allows for another layer? To answer this question we will build our own version of RLM and vary loss and the amount of time to adjust to find the optimal operating level.

BACKGROUND

In this section, we will explain the architecture and how it was implemented in our case. The Layered video on the server side is depicted in figure 1.

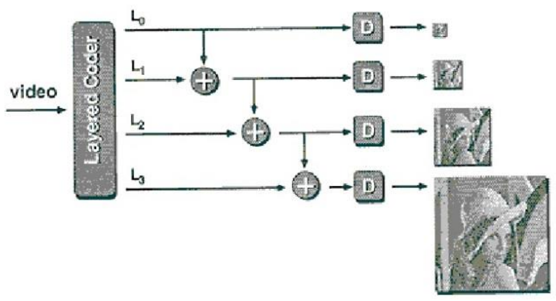

Figure 1. Layered video coder

L0 represents the base layer where the video is at its lower quality. As we add a layer the quality gets better and size gets bigger. The Client will adapt the quality received depending on the network congestion.

Since the server is multicasting the video, he cannot send specific data to each client, that's why we need a router in between. The router will receive all the channels that the server is multicasting and then unicast them to the player depending on what the player is asking. The architecture is depicted in figure 2.

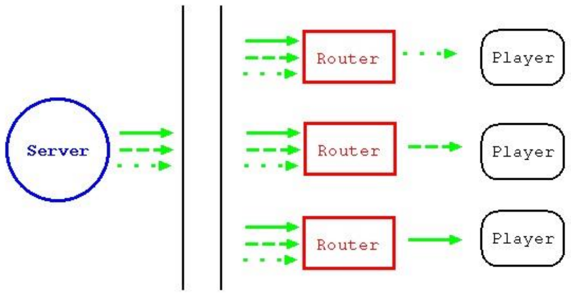

Figure 2. The relationship between the Server, Router and Player

In our case, the server will send three different qualities on three different channels. Each of the channels will have a different multicast address. The router who receives all of them will send only one, two or the three of them to the client depending on what he asks.

IMPLEMENTATION

As videos we are using text based videos and we have three different programs.

The Server: sends one frame per second for each of the channels. Channel 1 will have frames 1, 4, 7 ... Channel 2 will have frames 2, 5, 8 ... Channel 3 will have frames 3, 6, 9. That way the more bandwidth the client has, the more channels and frames he will get.

The Router: receives all multicast frames on all channels from the server, sends via unicast (UDP) frames of a specified quality level to the player and changes the quality level depending on the player's feedback.

The Client: The player receives at first all the frames from all the channels and playing them. The player employs the basic RLM protocol for quality ranges 1-3: Upon detection of a variable number of lost packets, the player assumes congestion and reduces the quality level. If a packet is received the player will send and "ack" to the router, if not the player will send "loss" to the router. After a certain number of "loss" received the router will drop a channel. After waiting for some time which means a certain number of consecutive "ack" sent to the Router, this means no more loss is detected and the router will send another channel and thus increase the quality level.

PROCEDURE

Now to determine when should the Player drop a channel and how long does he need to wait to add a channel, we will run an experiment having different amount of "loss". The loss will be simulated at the Router level when instead of sending a frame he will send "loss". The amount of loss will be randomly generated with different amounts. We will have three variables to vary.

1. Vary loss from 5% to 10% to 15% to 20%.
2. Vary when to drop a layer, after 1 lost packet or after 2 lost packets (waiting time will be 4).
3. Vary waiting time to assume that network is ok, after 2, 3 or 4 consequent packets arriving (drop after 2 packets lost).

The experiment will be run as follow: For each loss, we vary when to drop a layer and then do the same for waiting time. For this experiment, we are using text-based video starwars.wav which contains 82 frames. Since we are using text-based video at 1 frame per second, it's hard to assess the quality, so for the experiment the participants will watch the video without any loss first, and then watch the video with different losses and variables. We will ask them to rate the video from using MOS (1-5) assuming that the original video is a 5, that way we will avoid lower rates due to the nature of the video. We will also ask the participants to pick a favorite video session considering one percentage of loss and one variable changing. Example: For 5% of loss, which video was your favorite the first (drop after 1 packet) or the second (drop after 2 packets) or was there no difference?

After collecting and analyzing the data, we will deduct an optimal setting for the two variables that gives the best experience to the client.

RESULTS

The first experiment consisted on comparing two approaches dropping a channel after 1 loss (conservative) and dropping after 2 losses (less conservative). The results are shown in figure 3.

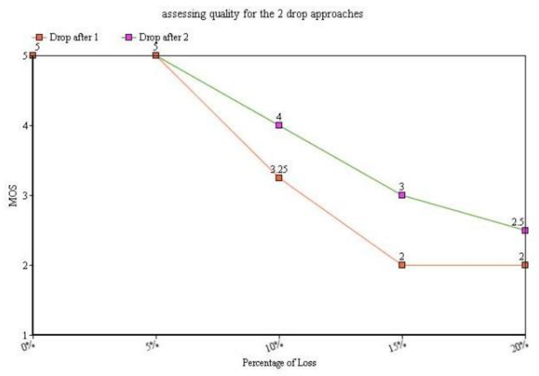

Figure 3. Comparing the 2 drop channel approaches for different loss percentage

	Drop after 1	Drop after 2
5% loss	0	0
10% loss	0.5	0.81
15% loss	0.81	1.41
20% loss	0.81	0.57

Standard deviations

The results show that less conservative approach has better score for all the percentages of loss which is also expectable since we wait to be sure that there is congestion before dropping a channel and that way we keep a good quality as long as possible. The figure 4 that shows which approach was preferred confirms these results. However, for 5% loss users see no difference in the video. Also for 20% loss users find the two approaches almost similar.

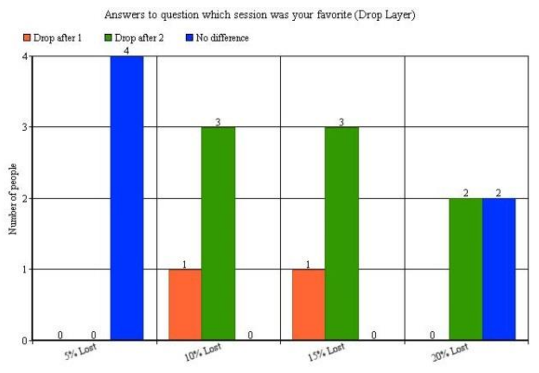

Figure 4. Favorite drop approach according to users

The second experiment consisted on comparing the three approaches of adding a channel. Each one represents after how many consecutive packets received should we consider adding a channel.

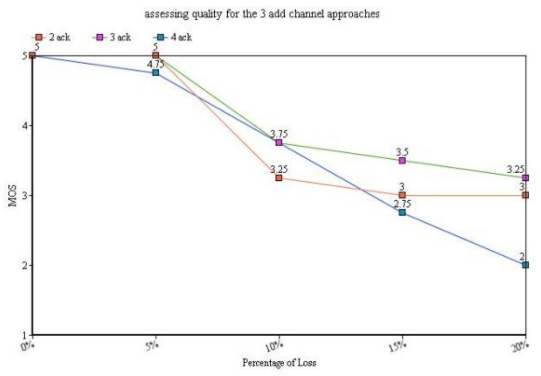

Figure 5. Comparing the 3 add channel approaches for different percentages

	2 “ack”	3 “ack”	4 “ack”
5% loss	0	0	0.5
10% loss	0.5	0.5	0.5
15% loss	0.81	0.57	0.5
20% loss	0.81	0.5	0.57

Standard deviations

For the 2 “ack” and the 4 “ack” approach the results are overlapping we can’t really tell which one is better. However, 3 “ack” have always either similar or slightly better results than the two others. The advantage of 3 “ack” is not really obvious especially while looking at figure 6 which shows the favorite video session for the users. 3 “ack” in green doesn’t really obtain a clear preference from the users.

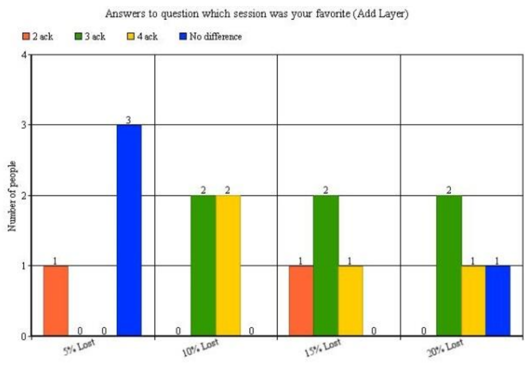

Figure 6. Favorite add approach according to users

ANALYSIS

From our results, we could deduce that change in the video quality shouldn’t occur too fast. In fact, dropping a channel after only one loss and adding a channel after only 2 consecutive packets received had somehow the worst results according to the user study. From the results, we could also determine which variables provide the user with the optimal operating level. Those variables are: dropping a channel after 2 lost packets and adding a layer after 3 consecutive packets received.

CONCLUSION

After building our Mini-RLM program and testing it with users under different loss rate and parameters, we were able to determine an approximate set up that will provide an optimal operating level to the user. However, that set up would be better if it actually adjusted to network conditions, for example be more conservative when there is too much loss and be less conservative when there is not that much. For future work, we propose to build an application that uses real video and more channels so that testing results will be more significant and feedback from users could be more useful.

REFERNCES

[1] S. McCanne, V. Jacobson, M. Vetterli Receiver-driven Layered Multicast, August 1996