20 February 2016

UBC CPSC 416 Distributed Systems: Project Proposal

Alcuaz, Ito Franchilo Mikael[1] — Aziz, Shariq[2] — Ko, Mimi[3] — Musa, Abrar[4]

Abstract

Media streaming refers to the constant delivery of time-ordered multimedia content from a server to a client. It is an attractive option as opposed to conventional file transfer methodologies as it provides instant access for the end user and is quite flexible – a user doesn't have to download the entire file beforehand as was traditionally done in the past, and can interact with the data as it arrives. Naturally, such a system demands a high bandwidth connection and so works more effectively for users on faster internet networks. This proposal serves as an outline for a possible solution to this key shortcoming by de-centralizing the server-client model through the use of peer-to-peer (P2P) stream networking.

[1] *y9u8, ialcuaz@alumni.ubc.ca*
[2] *i2u9a, shariqazz15@gmail.com*
[3] *o3d7, mimi@dbzmail.com*
[4] *i1u9a, abrar.musa.89@gmail.com*

Contents

1. Introduction

A P2P implementation of a media streaming system alleviates some of the bandwidth requirements due to the fact that content is distributed over several streams rather than over one singular server. Usually the content is multicasted from a server to some number of clients which has requested the stream; YouTube makes use of this notion by deploying multiple content delivery networks (CDNs) which distribute the content by demand; the video content is streamed over a single connection, and as such, demands a high bandwidth for maximum playability.

The idea behind using P2P streaming is to make the system more scalable – as the number of clients increase, so do the number of potential peers which can now act as a source, and thus share part of the bandwidth requirements. It does this by making the data available to other clients who also requested the stream, similar to how P2P BitTorrent file sharing works. The server and clients therefore together form a network of media streams.

Such a system, however, must preserve an invariant trait: clients must be able to connect and disconnect at will – it must be able to discover the parent server through a given URL and leave at any point in time; client behaviour is inherently unpredictable especially in P2P systems which presents numerous complications. There must also exist logic to decide which clients copy parts of the stream, and application level functionality to allow a client to reject P2P connections (similar to rejecting seeding in BitTorrent applications).During the early 2000s, studies have been done on the usage of overlay systems for multicast applications. These systems were conventionally implemented at the end systems.

A few popular P2P implementations are PeerCast (2004) [1], SplitStream (2003), SCRIBE (2002), and Pastry (2001) [2]. The first in this list will be examined primarily and used as a model for this group's implementation which will use the Go programming language.

2. Implementation Strategy

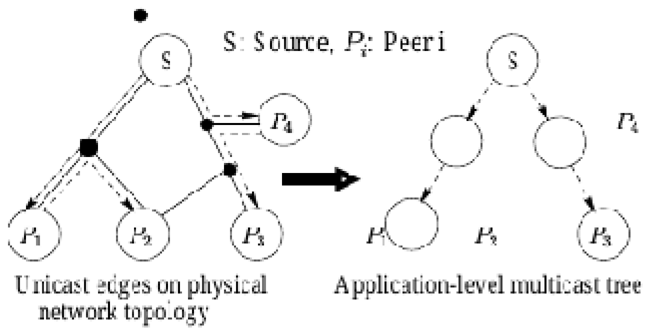
S: Source, P_i: Peer i
S
P_4
P_1
P_2
P_3
Unicast edges on physical network topology
Application-level multicast tree

Figure 1. A multicast tree.

2.1 Software engineering practices

To follow good industry practices, this group will probably follow test driven development (TDD) and SCRUM methods throughout the development of the project, so that feature implementation is planned and executed with minimal risk.

2.2 Construction of a single, centralized server, and requesting clients

This is the base case of the problem statement and should be implemented first and foremost. A server with a dummy file will be set up and a client will connect and stream the file to completion through a connection. To closely mimic standard video transferring protocols, a TCP connection will be first set up between the two on the initial handshake phase, and then use UDP packets to stream the data. It is critical that the packets be time-ordered. The server should be able to handle an arbitrary number of clients, and detect disconnections and failures. The dissemination tree of a basic multicast overlay system is thus constructed at this step.

2.3 Stream forwarding policies

Clients must be able to replicate the stream or parts of it from the server. An algorithm to decide where the server forwards its stream will also be implemented; the amount of forwarding must scale proportional to the number of connected clients. As above, failures and disconnects must be detected, such as the case of a replicated stream that is lost. In this step the basic join/leave/stream transfer policies as outlined in SCRIBE/Pastry will be implemented.

2.4 P2P streaming, edge case failures, and application level functionality

Clients will then be able to stream the data through other clients; there must exist an algorithm to decide when to do so, and the client which is streaming data to another client must communicate with the server to avoid data redundancy. This is the point in the project where the discussion, edge cases, and some of the specific heuristics discussed in [1] and [3] will come into fruition, and as such, will likely take significantly more time than the other segments. It would also be a nice plus to implement a feature which allows a client to prevent itself from acting as a source.

2.5 Project deployment and extras

In this section the GUI will be implemented, and the project will be hosted on a free website, possibly Heroku. It is hoped that any user can then upload multimedia content to some centralized server (maybe their machines can also act as servers ?), of which other users can then stream using the algorithms implemented above.

3. Project Timeline

To allow as much time as possible for polishing, debugging, and unforeseeable events, the meat of sections 2.2 - 2.4 should be completed on, or 1-2 weeks after Mar. 18, which corresponds to the project status meeting date. Step 2.2 most definitely needs to be finished as soon as possible within the next few weeks and shouldn't take as long as the subsequent steps.

4. SWOT Analysis

4.1 Strengths

There is a wealth of resources available on P2P streaming and established solutions which would serve to simplify the design and implementation process. The P2P system self scales as more nodes join, which is at the heart of distributed systems. PeerCast and other similar systems are well studied with respect to their advantages, drawbacks, and vulnerabilities.

4.2 Weaknesses

The system will be difficult to test and complicated to build due to the replication procedures, P2P links, unpredictability of node behaviour, etc. Performance will also be difficult to optimize as opposed to unicast-esque server-client implementations.

4.3 Opportunities

The Go programming language was created to specifically build distributed systems. As was seen in the class assignments, standard networking procedures are fairly straightforward to implement using this language. This project can be easily integrated with newer ideas r.e. media streaming.

4.4 Threats

The Go programming language is fairly new which makes well established solutions to specific problems harder to come by than other languages. Furthermore, the members of this group have not had experience with it prior to taking this course, and as such, the best practices and idioms that come with this language will not come as intuitively.

5. Conclusion

A possible solution to the drawback of the centralized client-server model in media streaming can be found through the use of a multicasted, P2P network. This is achieved by sharing the bandwidth requirements across several disjoint nodes which serve to forward the data stream among themselves, not over one singular connection. Such a system is complicated to build however, and gives rise to its own set of disadvantages.

References

[1] H Deshpande. Streaming live media over peers. http://ilpubs.stanford.edu:8090/863/1/2002-21.pdf, 2002. Accessed: 2016-20-02.

[2] A Rowstron. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. http://research.microsoft.com/en-us/um/people/antr/PAST/pastry.pdf, 2001. Accessed: 2015-20-02.

[3] Z Shen. Peer-to-peer media streaming: Insights and new developments. http://www3.ntu.edu.sg/home/junluo/documents/P2PSurvey.pdf, 2011. Accessed: 2015-20-02.