

Assignment 2

CS414, Multimedia Systems (Instructor: Klara Nahrstedt)

Posted: February 9, 2009

This is the second assignment for your Peer-to-Peer Video-on-Demand (P2P-VOD) project where you start to build the network and P2P fabric of your video-on-demand (VOD) server with your group members. This assignment will have three goals: (a) bootstrap the p2p-vod system, i.e., build the network connectivity among the peer servers themselves and the dispatcher server, and build the network connectivity between the p2p-vod client and the p2p-vod service, (b) build distributed operations on top of the network fabric implementing operations such as INSERT a video file, INSERT an audio file from client to the VOD service and DELETE a file in the VOD server(s). All assignments will be implemented on the Linux Dell Machines in 0216 SC lab.

Assignment Description:

1. P2P Video-on-Demand (VOD) Server Architecture

The P2P-VOD Server will consist of three peer servers (S1, S2, S3), and one dispatch node (dispatcher, also called front-end engine) as shown in Figure 1. The dispatcher role will be to receive any requests from the clients to operate over the P2P-VOD server. The peer servers, e.g., S1, S2, S3 will carry the video and audio files (content).

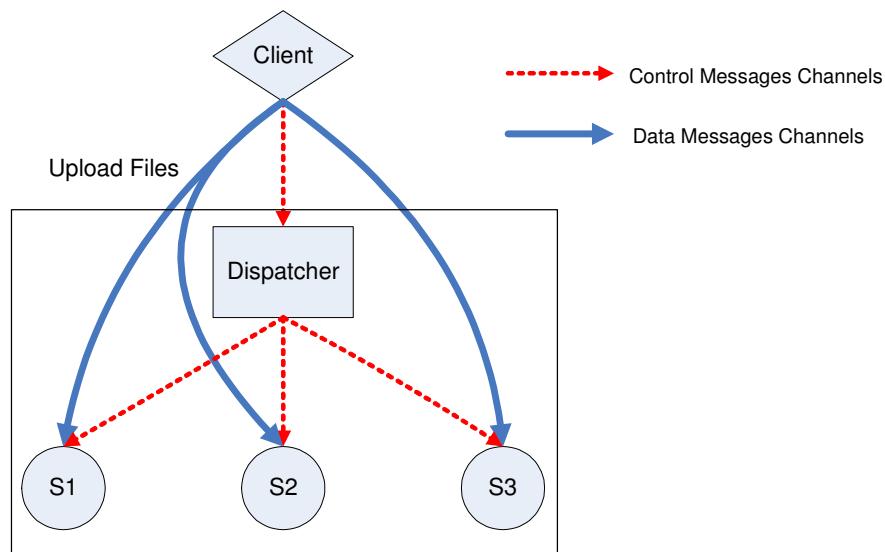


Figure 1: p2p-vod server architecture where S1, S2, S3 are the peer-servers of the p2p-vod server to keep the server content A/V files and the dispatcher is the front-end central engine (like a super-node in Napster) of the p2p-vod server to keep state information about the participating peer-servers.

Client is outside of the P2P-VOD server architecture and when it wants to insert a file, it has to first contact the dispatcher to find out on which peer-server the content should reside in case of insertion. Once the client knows which peer-server will be responsible for the file insertion, the client and the peer-server establish a connection and upload (insert) the A/V file.

Data Plane Functions		Control Plane Functions
Stream (Play, FF, Rewind) A/V Files	Insert (Upload) A/V Files	Insert/Delete/Search Stream Request Control Protocols and Services
Setup UDP Connection between Client and a Specific Peer-Server	Setup TCP Connection between Client and a Specific Peer-Server	Dispatcher setup Peer-Server addition/Connection setup between Dispatcher and Peer-Servers for Control Purposes
UDP/IP Protocol Stack	TCP/IP Protocol Stack	TCP/IP Protocol Stack

Figure 2: Protocol Architecture on Client/Peer-Servers/Dispatcher to implement the P2P-VOD Functions

To achieve this type of architecture, in this MP2, you need to build step by step various protocols on top of the TCP or UDP/IP network protocol stack as shown in Figure 2.

2. Building Control Plane for P2P-VOD Service

2.1. Establishing Connectivity in p2p-vod Server (Bootstrapping the system)

To bootstrap the p2p-vod system, you need to execute the following steps:

1. **Start and Setup the dispatcher** – dispatcher is a special control node that will host the state information about the A/V files at the participating peer-servers.
2. **Start and Add peer-servers** – You should setup TCP/IP connections (red lines in Figure 1) between the dispatcher and peer-servers. Make sure that the adding of peer-servers is dynamic in the sense that one can add 1, or 2 or 3 or more peer-servers during the demonstration. These control connections will be used to transport control messages among dispatcher and peer-servers to insert a A/V file into the p2p-vod server, to delete A/V file from the server, and to find A/V file in the p2p-vod server. The following issues should be considered:
 - a. In order to establish this connectivity, you need the **peer-servers (S1, S2, S3)** ‘register’ with the dispatcher so that the dispatcher can keep a **membership list** of all peers that report to it.

- b. In order for a client to start a service with the p2p-vod server, the client needs to connect **to a front end node** of the p2p-server, which in our case is the dispatcher.
3. **Connect client to dispatcher** – you should setup TCP/IP connections between the dispatcher and the client which wants a service from p2p-vod server.

2.2 Implementation of Insert Function

To build the insert function, it will have to major components: (a) control plane function and (b) data plane function. In this subsection we will only explain the control plane function of “INSERT”. In the following section we will then discuss the data plane function of “INSERT”. In the control plane function, you need to establish the following protocols (note that this is a proposal, you can change the protocols, i.e., message flow, as long as you provide the main functionality of “INSERT”).

- a. Client **sends an “INSERT” request** over the control connection to the dispatcher with the information (client-name, file-name, “INSERT-op”).
- b. The dispatcher **establishes a lookup (index) table**, if this is the first “INSERT” request.
- c. The dispatcher **matches the “INSERT” request** with the peer-server (S1, or S2, or S3) in the membership list according to a **load balancing algorithm**.
The load-balancing algorithm should make sure that only 8 files are on each server (independent if they are audio or video files). This means that, for example, if there are already 8 files in S1, the dispatcher must insert the file in S2 or S3 if they have less than 8 files on their server.
- d. The dispatcher **makes an entry into the lookup table** with information (***File-name, Peer-Server-name, Flag***), where ***file-name*** is the name of the inserted A/V file (e.g., movie.mjpg, or song.au), ***Peer-Server-name*** is the name of the peer-server where the A/V file should be inserted, and ***Flag*** is the flag that indicates if the “INSERT” operation has been completed.
- e. The dispatcher **sends a response** to the client with the selected ***‘peer-server-name’*** (e.g., S1).
- f. The dispatcher **sends a message to the selected peer-server** (e.g., S1) to inform it that it should expect a connection request from a client with ***‘client-name’*** and an A/V file with ***‘file-name’***.

2.3 Implementation of Delete Function

To build the delete function of the p2p-vod service, there will be only a control-plane functionality/protocol between the client/dispatcher/peer-servers as follows:

- a. Client **sends “DELETE” message** request over the control connection to the p2p-vod dispatcher with the information (***client-name, file-name, DELETE-op***).
- b. Dispatcher **finds the (*client-name, file-name*)** information in the look-up table and finds the ***peer-server-name*** (e.g., S1).
- c. Dispatcher **sends a “DELETE” message** request over the control connection to the selected ***peer-server-name*** (e.g., S1).
- d. The peer-server **deletes the file and sends a “OK” message** to the dispatcher.

- e. The dispatcher **deletes the entry** from the look-up table.
- f. The dispatcher **sends “OK” response** to the client.

3. Building Data Plane for p2p-vod Service

In this MP2, you will build the data plane functionality of the “INSERT” function to allow a client to **upload A/V file** into the peer-server, selected by the dispatcher. You may follow the following protocol:

- a. Client should have at this point the *peer-server-name* from the dispatcher (see Section 2.2).
- b. Client establishes a new TCP/IP connection between client and the selected peer-server (at this point the peer-server should know and wait for the connection establishment request – see Section 2.2)
- c. Once the client establishes the connection, it uploads the audio or video file into the server in chunks – like an ftp service. At this point you don’t need to worry about streaming the audio or video from the client to the server. You can choose big chunks of data from the audio or video file to upload it as fast as possible.
- d. Once the A/V file is uploaded, the client should cleanly close the connection to the peer-server.
- e. Client (or the selected peer-server) should inform the dispatcher that the upload operation was successful (over the control connection between the client and the dispatcher).
- f. Dispatcher should make a note in the lookup table that the INSERT operation was completed successfully (e.g., you could have a FLAG associated with each entry in the lookup table that would have value 0 or 1, meaning 0 – incomplete, 1 complete operation)>

4. P2P-VOD Interfaces

- 1. You will need to start the peer-server programs individually, as well as start the dispatcher program at their individual nodes. Then you will need an interface to bootstrap the p2p-vod server. This interface will execute the functions/protocol discussed in section 2.1. An example of an interface could be
 - a. **‘Dispatcher S1 S2 S3 enter’**, where with this command you start adding the peer-servers to the dispatcher and building the membership list at the dispatcher.
- 2. You should implement an interface to call the individual distributed server functions, i.e., the INSERT/DELETE functions of the *p2p-vod* service from the Linux command line from the client. This interface will be implemented by the protocols in section 2.2-2.3. An example of a command line could be
 - a. **‘p2p-vod function file’** where the *‘function’* is at this point INSERT or DELETE. *‘file’* represents the *file-name* of the file you are aiming to insert, or delete.

Delivery

Each group delivers:

- source C, or C++, or Java of your **p2p-vod** program in the particular group directory. The source code evaluation will be based on how well is your code documented. If you use some code you found on the web (You must understand the code you found and include in your code, not just blindly copy the code !!!) or in a local system directories, document it. It is very important that you give credit to people who developed the previous code. Your own code should include the following information at the beginning of each C/C++ /Java file.
- Each major source file should include
 - File Name: Name of the File
 - Description: Short description what the file includes (general description, what kind of functions are embedded in the file).
 - Version: version of your code. You start with version 0 and as you improve the code, at some point you increase the version.
 - Programmer's Name: your name(s) who developed the code
 - Company/University Name: you put the name of the course, department and university you implemented the code for;
 - Date:
- Each function in your C/C++ /Java file should have a header with information:
 - Function Name: Name of the Function
 - Description: Short description what the function does.
 - Arguments: Specification of each input argument parameter entering the function and its description.
 - Results: Specification of returning parameters exiting the function and their description.
 - Comments: some special system issues connected with this function
- Group representative(s) comes at the scheduled time (we will have a sign-up sheet) between **5pm and 7pm on Monday, March 2nd** and shows a demo of the required programs in 0216 Siebel Center.

Evaluation of the Assignment (100 Points)

- **Bootstrap the p2p-vod service** (discussed in Section 2.1) with the functions: start dispatcher, start peer-servers, connect dispatcher and peer-servers (add peer-servers) and establish membership list, and connect client to the dispatcher - **20 Points** for demonstration
- **Bootstrap questions – 5 Points**
- **INSERT function** (discussed in Section 2.2) with capabilities: send 'INSERT' request to dispatcher, create or update index/lookup table, find if file exists, do load balancing, specify which peer-server will hold the file, return to client the peer-server name – **15 Points** for demonstration
- **Insert function in control plane questions – 5 Points**
- **INSERT function** (discussed in Section 3) with capabilities: Upload actual file from client to one of the peer-servers – **15 Points** for demonstration
- **INSERT function in data plane questions – 5 Points**
- **DELETE function** (discussed in Section 2.3) with capabilities: send 'DELETE' request to dispatcher, find entry in the lookup table, remove the entry at the dispatcher's lookup table, remove the file in the peer-server – **20 Points** for demonstration

- **DELETE function questions – 5 Points**
- **DOCUMENTATION - 10 Points** - Each group should write a short README file (1-2 page) in pdf format. In this documentation file you should specify your design/implementation document, short description of functionalities in terms of their implementation - don't repeat what is in this assignment write-up). Email this document to the TA hnguyen5@uiuc.edu, and also store it in your group directory.

The demonstration of the whole assignment for one group should take no more than 20 minutes.