Customized and manageable streaming service using a file-converting server

Seunghoon Jeong
UCSD CSE
Outline

1. Introduction
   - Background
   - Problems
   - Motivation

2. Design
   - Goals
   - Scenario
   - Model

3. Implementation
   - Client
   - Server
   - Dynamic control

Conclusion
Introduction
Background

- Cell phones and PDAs can now play movie clips and TV programs

- Increasing numbers of them are now equipped with web-accessible ability
Problems

- limited contents available
  - Incomparable to the amount of Internet video material

- Non-stream format video files
  - avi, mpeg, mov, ...

- low processing ability and small LCD size
  - Large-sized high quality video files for PCs

- Unable to control streaming data dynamically
  - Lack of supports by streaming servers
Motivation

- Powerful processing ability of PC
  - Encoding and Decoding in short time

- Superior wired network speed
  - More stable and faster than wireless network

- Various streaming protocols
  - Relief of file saving in client and support of streaming control

- Utilize a third party server!
Goals

- Enable non-stream format video files to be played
  - Delegate downloading to a server and let it service streaming

- Prevent wastes of resources
  - Customize streaming through optimizing process

- Provide dynamic management of streaming data
  - Make server to support seeking functionality
Scenario

1. Video clip URL
2. Redirected URL with cell phone information
3. Downloading the requested file
4. Conversion
   - ASF
   - AVI
5. Manageable streaming service

AVI

AVI
Model
Implementation
Client Implementation

- User command processing module
  - Interface of file and stream control, conversion info setting

- Mplayer wrapping module
  - forks mplayer, pipelined communication with mplayer

- MPlayer
  - slave mode running, stream controlling

- Communication module
  - TCP/IP based, several protocols and data types defined
Server Implementation

- Communication module
  - Controlling multiple clients

- Downloading and converting module
  - ‘wget’ for downloading, ‘ffmpeg’ for converting

- Streaming module
  - mplayer/libmpdemux/netstream
  - mplayer/TOOLS/netstream
Dynamic Controlling of a stream

Moving
10 sec forward

User command processing module

1. Move 10 sec Forward

[commands by the client]
NET_STREAM_OPEN: request to start streaming service
NET_STREAM_SEEK: request to reset streaming position
NET_STREAM_CLOSE: request to finish streaming service
NET_STREAM_RESET: reset the current streaming service

[server responses]
NET_STREAM_OK: command applied successfully
NET_STREAM_ERROR: failed to apply command

Mplayer wrapping module

2. 'Seek 2 10m'

Streaming module

5. mp_net_stream_packet_str
   [NET_STREAM_SEEK, 10]

Communication module

[communication module]

Client

Internet

Downloading and converting module

3. mplayer.c.
   stream_seek(ths, 10)

4. stream_netstream.c.
   send_net_stream_cmd(10)

6. netstream.c:
   handle_client(client_pkt)

7. netstream.c:
   netstream_seek(client_pos)

8. mp_net_stream_opened_str
   [size_start_pos(10), end_pos]

[database structure sent by the client]
struct mp_net_stream_pack_st {
  int len,
  int cmd,
  char[] data
}

[database structure sent by the server]
struct mp_net_stream_opened_st {
  int file_format,
  int flags,
  int sector_size,
  int start_pos,
  int end_pos
  char[] data
}
The client could play a large sized non-stream file
  - without using NFS and client’s need to save files

Waiting time to play a movie file was greatly reduced
  - downloading time shortened
  - file size reduced by optimizing process

Dynamic control enabled
  - FF/REW. skipping, resizing