Project Proposal of CS641: Video Streaming Over Peer-to-Peer Network

Xu-Xiang JIANG and Yu DONG

Sep,2002

1 Introduction

Peer-to-peer system has recently gained popularity thanks to the emergence of file sharing applications over the Internet. In a peer-to-peer system, users are at the same time users and providers of information, which is usually exchanged through pairwise interactions. Whatever definitions have been put upon it, peer-to-peer is an effective rallying cry for a new way of doing things. Streaming media delivery is particularly susceptible to a peer-to-peer architectural approach. In this proposal, we try to identify important research issues related to the streaming media delivery in P2P system. Also we plan to construct a real functional software which integrate the gnutella peer software with media-streaming support. Our approach for the P2P media streaming, basically, has two components: Network Layer and Media Player User Interface. Once the user want to view the media file, the Network Layer will: firstly, locate sites containing the media and secondly stream from them. The way to stream should distribute the network load evenly or proportional to the peer capacity(computing power/ network connectivities). Media player User Interface need to be adaptable in the sense of cooperating with Network Layer and avoid the jitter as much as possible.

2 Related Works

Peer-to-peer system isn't a new concept, but recently it gains tremendous attention not only from the research community, but also from the industry thanks to the availability of existing peer-to-peer systems, such as Napster[5](which is defunct now due to the lawsuit), gnutella[3], eDonkey[6] etc. The Microsoft .NET architecture claims to have the support of peer-to-peer features [7]. But just like [1] mentioned: one category of peer-to-peer system has so far received less attension: the peer-to-peer media streaming system. SpeedIt [9] is an architecture proposed for streaming live media over peer-to-peer network. But it requires synchronous streaming among a group of users and don't deal with bandwidth adaptability and admission differentiation. C-star[4] is a commercial multi-source streaming service, but also it doesn't differentiate suppliers which have network connectivities. In [8], a distributed video streaming is also presented, but essentially it's still a client-server system instead of P2P system so it suffers from common deficiencies of central server architecture: system scalability problem.

3 Problems we will exploit

We need to address the following problems

- How to locate the media?
  Different ways to locate the media can have different impact to the underlying system. Also it's maybe constrained by the underlying overlaying architecture. For example, the Napster[5] used one central database to store the mapping between file and sites containing the file. We have to check the central server first to locate the sites. Obviously this approach is easy but incurs burden for database, and not scalable. The requirement for locating process should be efficient, scalable and robust.

- How to retrieve the media?
  Some peer-to-peer systems use caching and replications strategies to speed up the retrieval process. Chord [2] caches the blocks just retrieved in the lookup-path nodes Pastry replicates the whole files in the neighbor nodes. The requirement is low time delay, and small retrieval overhead.

- How to play the fetched media?
  Due to the varying network load and potential congestion, the way to play the media should allow different network conditions. And it's intolerable to view distorted or jittered video. One common approach is using the buffer mechanism.

In out plan, for the locating problem, we'll focus on the widely used Gnutella system. We would identify the locating mechanism used in the Gnutalla. For the retrieving problem, firstly, we try to download from one single source and then extend to multi-source if time permits. For the video streaming playback, we will firstly concentrate on the MPEG format video since it's one of the most popular video coding format and there would be large amount potential video sources available on the Internet. Also, as one of our future research interests, we will exploit other video format to extend our system to provide QoS service through hetergenous Peer-to-Peer network.

Most of the related works as ?? use the commercial media player such Microsoft MediaPlayer, Apple QuickTime and RealPlayer. This not always desirable since the users have to install such a *large* player on their hosts which sometimes has limitation of the resources avalible. Our implementation tries to provide one kind of light-weighted system which could also be used on resource-limited host.

Since our system will playback the video stream transported through Internet instead of the traditional video file, it will need a more dedicate buffer control which behaves as the interface between the underlying overlay network and the player. For this buffer control, we need to fulfill the following functions

- managing overflow and underflow of video stream
- demultiplexing the video data only(put this into underlying overlay network module is also alternative)
- sending control messages to both overlay network and player of the current data stream information to request next data stream from overlay network and provide the available data stream to the player module.

If time permits, we'll also support to retrieve the same video stream from multiple peers to pre-fetch data, and thus to achieve more reliable effect of playback. This will require to extends the functions of buffer controller to support such multiple retrieving data stream.

To design the light-weight player of our system, we will take advantage of some available free-distributed source library of MPEG codec. However, most of these libraries don't support streaming and only provided limited function, plenty of work also need to be done to adapte it to our system. Such as changing the interface of codec to integrate with buffer controller, transforming the YUV output format to default RGB format to display on the platforms as Windows.

4 Implementation Schedule

The following is roughly coarse time planning:

- 9.28 - 10.5 survey of related works and feasibility, problem identification
- 10.5 - 10.10 Division of responsiblities
- 10.10 - 11.10 prototype implementation individually
- 11.10 - 11.30 system intergration and final report

References

[1] D. Xu and M. Hefeeda and S. Hambrusch and B. Bhargava, "On Peer-to-Peer Media Streaming", *Purdue Computer Science Technical Report*, Apr. 2002.

[2] Ion Stoica and Robert Morris and David Karger and Frans Kaashoek and Hari Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications", *Technical Report TR-819, MIT* , March 2001.

[3] Gnutella *http://gnuttella.wego.ocm*

[4] C-star *http://www.centerspan.com*

[5] Napster *http://www.naspter.com*

[6] eDonkey *http://www.edonkey.com*

[7] Microsoft MSDN *http://msdn.microsoft.com/msdnmag/issues/01/02/netpeers/netpeers.asp*

[8] T. Nguyen and A. Zakhor. Distriubted Video Streaming Over Internet. *Proceedings of SPIE/ACM MMCN 2002*, Jan 2002

[9] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over a Peer-to-Peer Network. *Stanford Database Group Technical Report (2001-30)* Aug. 2001