Peer-to-Peer Streaming

Wolf-Tilo Balke and Wolf Siberski

23.1.2008

*with slides from Kan-Leung Cheng (U Maryland), Darshan Purandare (UCF), Hui Zhang (CMU), Long Vu (UIUC)

Overview

1. IP-Level vs Application-Level Streaming
2. P2P Streaming Basics
3. PPLive
4. Summary
Multicast Internet Applications

- **Multi-party applications**
  - Video on Demand and IPTV
  - Audio/video conferencing
  - Multi-party games
  - Distributed simulation
  - Broadcast of web cams

- **Consider a world with ...**
  - Tens of millions of simultaneously running multi-point applications
  - Each application with tens to several thousand of end points

---

**Multicast vs. IP Multicast**

[Diagram showing comparison between Unicast and IP Multicast]
IP Multicast Overview

- Seminal work by Steve Deering in 1989
  - Huge amount of follow-on work
  - Research
    - 1000s papers on multicast routing, reliable multicast, multicast congestion control, layered multicast
    - SIGCOMM, ACM Multimedia award papers, ACM Dissertation Award
  - Standard: IPv4 and IPv6, DVMRP/CTBT/PIM
  - Development: in both routers (Cisco etc) and end systems (Microsoft, all versions of Unix)
  - Deployment: Mbone, major ISP’s
  - Applications: vic/vat/rat/wb…

- Situation today
  - Still not used across the Internet
  - Reasons?

Router state issue

- How to tell a packet is a multicast packet?
  - Each group needs a group address

- How to decide where and how to branch?
  - Routing protocol needs to set up per group state at routers

- Violates stateless packet forwarding principle
  - Currently IP layer only maintains routing state
    - Highly aggregated
  - 140K routing entries today for hundreds of millions hosts
**Ack Explosion**

- Large number of acknowledgements required
  - End-to-end acknowledgments inefficient
  - Router-based acknowledgments overloads routers
    - requires even larger state maintenance

**IP Multicast Issues – Summary**

- Poor routing scalability
  - Routers need to keep per group/connection state
  - Violation of fundamental Internet architecture principle
- Difficult to support higher functionalities
  - Error control, flow control, congestion control
- Security concerns
  - access control, both senders and receivers
  - Denial of Service attacks
IP Architecture

- "Dumb" IP layer
  - minimal functionalities for connectivity
  - Unicast addressing, forwarding, routing
- Smart end system
  - transport layer or application performs more sophisticated functionalities
  - flow control, error control, congestion control
- Advantages
  - accommodate heterogeneous technologies
  - support diverse applications and decentralized network administration

Multicast revisited

- Can we achieve
  - efficient multi-point delivery
  - without support from the IP layer?
Application Layer Multicast (ALM)

- Do stream distribution on application level
  - Multicasting implemented at end hosts instead of network routers
  - Nodes form unicast channels or tunnels between them
  - Use default IP infrastructure
  - Receivers form self-organized network (peer-to-peer principle)

P2P Media Streaming

- Media streaming extremely expensive
  - 1 hour of video encoded at 300Kbps = 128.7 MB
  - Serving 1000 users would require 125.68 GB

- Approach: same idea as in P2P file sharing:
  - Peers form overlay network
  - Nodes offer their uplink bandwidth while downloading and viewing the media content
  - Takes load off the server
  - Scalable
Peer-to-Peer Streaming Benefits

- Easy to deploy
  - No change to network infrastructure
- Programmable end-hosts
  - Overlay construction algorithms at end hosts can be easily applied
  - Application-specific customizations
    - Network structure
    - Packet forwarding strategies

Challenges

- Need to playback the media in real time
  - Quality of Service
- Procure future media stream packets
  - Needs reliable neighbors and effective management
- High “churn” rate – Users join and leave in between
  - Needs robust network topology to overcome churn
- Internet dynamics and congestion in the interior of the network
  - Degrades QoS
- Fairness policies extremely difficult to apply
  - High bandwidth users have no incentive to contribute
  - Tit-for-tat doesn’t work due to asymmetry
Peer-to-Peer Streaming Models

- Media content is broken down in small pieces and disseminated through the network
  - Push model: content forwarded as soon as it arrives
  - Pull model: nodes request missing pieces

- Neighboring nodes use Gossip protocol to exchange buffer information
- Nodes trade unavailable pieces
- Robust and scalable, but more delay

Network efficiency

- Optimization Goals
  - Delay between source and receivers should be small
    - Relative Delay Penalty (RDP)
  - Number of redundant packets on any physical link should be low
    - Physical Link Stress (PLS)

High latency  High degree (unicast)  "Efficient" overlay
**Physical Link Stress (PLS)**

- PLS is given by the number of identical copies of a packet that traverse a physical link
  - Indicates the bandwidth inefficiency

- Example:
  - PLS for link S-R1 is 2.
  - Average PLS is 7/5.

**Relative Delay Penalty (RDP)**

- RDP is given by the ratio of the delay in the overlay and the delay in the direct unicast path.
  - Indicates the delay inefficiency

- Example:
  - Overlay delay for the path from S to E3 is 60 ms.
  - Unicast delay is 40 ms.
  - Therefore, the RDP for E3 is 1.5 ( = 60 ms / 40 ms).
Network topologies

- **Tree Based**
  - Content flows from server to nodes in a tree like fashion
  - One point of failure for a complete subtree
  - High recovery time
  - Multiple trees for increased robustness

- **Mesh Based**
  - Overcomes tree based flaws
  - Nodes maintain state information of many neighbors
  - High control overhead

Streaming Topologies: Tree

- Tree construction based upon minimal delays
- Permanent peer monitoring for tree maintenance and repair
  - In case individual links/path fail or become congested
  - In case receivers join and leave (churn)
- Push-based delivery from the source(s) to the receivers
  - Data forwarded along the tree with minimal delay
- Multiple trees (=multiple root nodes)
  - Provide redundant delivery paths against failures (churn)
  - Provide complementary data flows to each receiver
- Issues
  - Maintaining an “optimal” tree structure incurs a lot of overhead
  - Particularly in conjunction with churn
  - Churn may also cause disruptions to downstream receivers
Tree-based System: End System Multicast (ESM)

- First application level streaming system
  - Developed at CMU by Hui Zhang et al. (2002)

- Objectives
  - Self-organizing: adapts to dynamic membership changes
  - Self-improving: automatically evolves into efficient overlays

- Two versions of protocol
  - Multi-source, smaller scale conferencing apps
  - Single source, larger scale broadcasting apps

- Tree-based, Push model

ESM Node Join

- Bootstrapping process (for node X)
  - Connect to source (S)
  - Get a subset of group membership

- Parent selection algorithm
  - Send probe message to known nodes
  - Decision criteria for parent node (P)
    - Filter out P if it is a descendant of X
    - Performance of P
    - Delay of path from S to P
    - Saturation level of P
    - Performance of link P-X
    - Delay of link P-X
    - TCP bandwidth of link P-X
**ESM Tree Maintenance**

- Build separate control structure decoupled from tree
  - Each member knows small random subset of group members
  - Information maintained using gossip-like algorithm
  - Members also maintain path from source

- Continuously apply parent selection strategy
  - React on nodes becoming unavailable
  - Repeat probing regularly
  - Improve bandwidth usage
  - Improve clustering

---

**Multiple Tree Construction**

Diagram showing multiple trees with video stream.
Join Procedure

- Initial join
- Contact video source
- Receives peer list, number of trees
- Probe peers
- Connect to multicast trees

Disconnect / Rejoin Procedure

Yellow tree is down?

Yellow tree is recovered
Streaming Topologies: Mesh

- Mesh construction and maintenance similar to Tree
  - Find set of peers with minimal delays
  - Choose subset of the peers initially provided via bootstrapping
  - Gossiping protocols to learn about further peers
  - Continuous optimization of neighbor set
- Active pulling of media segments from peers
  - Exchange of buffer maps (who has which data)
  - Explicit requests for missing chunks (receiver-controlled)
  - Kept locally available for forwarding to other peers
  - Similar to BitTorrent, but needs to consider time-constraints
- Issues
  - Lots of control traffic (explicit pull)
  - Higher end-to-end delay (due to buffering and pull-based forwarding)

CoolStreaming

  - Joining Node obtains a list of 40 nodes from the source
  - Each node contacts these nodes for media content
    - Extend list of known nodes via gossiping
  - In steady state, every node has typically 4-8 active neighbors
  - Real world deployed and highly successful system
    - Stopped in 2005, due to copyright issues
    - Successor: PPLive
Coolstreaming distribution algorithm

- Stream is chopped by server and disseminated
- Each node periodically shares its buffer content map with neighbors
  - Request segments as part of this exchange message
- Reply strategy
  - Send scarce packages first (like BitTorrent)
  - If no package is scarce: send to peer with highest bandwidth first

Quality Criteria
### Quality Criteria

<table>
<thead>
<tr>
<th>Quality of Service</th>
<th>Jitter less transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low end to end latency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network efficiency</th>
</tr>
</thead>
</table>

| Uplink utilization | High uplink throughput leads to scalable P2P systems |

| Robustness and Reliability | Churn, node failure or departure should not affect QoS |

<table>
<thead>
<tr>
<th>Scalability</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determined in terms of content served (Share Ratio)</td>
</tr>
<tr>
<td>No user should be forced to upload much more than what it has downloaded</td>
</tr>
</tbody>
</table>

### Quality of Service

<table>
<thead>
<tr>
<th>QoS is the most important metric</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Jitter: Unavailability of stream content at play time causes jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jitterless transmission ensures good media playback</td>
</tr>
<tr>
<td>Continuous supply of stream content ensures no jitters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Latency: Difference in time between playback at server and user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower latency keeps users interested: a live event (e.g. soccer match) can lose importance in crucial moments, if the transmission is delayed</td>
</tr>
<tr>
<td>Reducing hop count reduces latency</td>
</tr>
</tbody>
</table>
Uplink Utilization

- Uplink is the most sparse and important resource in the network
  - Sum of uplinks of all nodes is the load taken off the server
- Utilization = (Uplink used / Uplink Available)
  - Needs effective node organization and topology to maximize uplink utilization
- High uplink throughput means more bandwidth in the network and hence leads to scalable P2P systems

Scalability

- Serve as many users as possible with an acceptable level of QoS
  - Increasing number of nodes should not degrade QoS
- An effective overlay node topology and high uplink throughput ensures scalable systems
**Fairness**

- **Measured in terms of content served to the network**
  - Share Ratio = (Uploaded Volume / Downloaded Volume)

- **Randomness in network causes high disparity**
  - Many nodes upload huge volume of content
  - Many nodes get a free ride with no or very little contribution

- **Must have an incentive for an end user to contribute**
  - P2P file sharing system like BitTorrent use tit-for-tat policy to combat free riding
  - Not easy to use it in streaming as nodes procure pieces in real time and applying tit-for-tat can cause delays

---

**Existing Systems**
Overview of Existing Systems

- Most noted approach in recent years: CoolStreaming
  - PPLive and SOPCast are derivates of CoolStreaming
  - Proprietary and working philosophy not published
  - Reverse engineered and measurement studies released

<table>
<thead>
<tr>
<th></th>
<th>Push/pull</th>
<th>Tree/Mesh</th>
<th>Buffer</th>
<th>Playout Delay</th>
<th>Startup Delay</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPLive</td>
<td>Pull</td>
<td>Mesh</td>
<td>2 min</td>
<td>1 min</td>
<td>20 s-2 min</td>
<td>300-350 kbps, Res: 320 x 240 pixel</td>
</tr>
<tr>
<td>Coolstreaming</td>
<td>Pull</td>
<td>Mesh</td>
<td>2 min</td>
<td>1 min</td>
<td>1 min</td>
<td></td>
</tr>
<tr>
<td>Anysee</td>
<td>Push</td>
<td>Hybrid</td>
<td>40 s</td>
<td>20-30 s</td>
<td>20 s</td>
<td></td>
</tr>
<tr>
<td>SopCast</td>
<td>Pull</td>
<td>Mesh</td>
<td>1 min</td>
<td>1 min</td>
<td>1-5 min</td>
<td></td>
</tr>
</tbody>
</table>

PPLive Overview

- One of the largest deployed P2P multimedia streaming systems
- Developed in China
- Hundred thousands of simultaneous viewers
PPLive Membership Protocol

Client

Channel management servers

(1)

Membership Servers

(2)

An overlay

Peers in the same channel

(3)

PPLive analysis


- Analyzed:
  - Channel size variation
  - Node degree
  - Overlay randomness
  - Node availability
  - Session length
Channel Size Varies over a day

- Huge popularity variation
  - Peaks at noon and night
- Higher dynamics than P2P file sharing

Node Degree

- Degree independent of channel size
- Similar to P2P file sharing

Average node degree scale-free
PPLive Peers are Impatient

50% sessions are less than 10 minutes

- Short sessions (probably channel hopping)
- Different from P2P file sharing

Conclusions

- Characteristics of PPLive and P2P file sharing are different
  - Higher variance in item popularity over time
  - Shorter average session duration
    - Much higher network churn
Summary

Current Issues

- High buffering time for P2P streaming
  - Half a minute for popular streaming channels and around 2 minutes for less popular
- Some nodes lag with their peers by more than 2 minutes in playback time
  - Better peering strategy needed
- Uneven distribution of uplink bandwidths (unfairness)
- No consideration of duplicate packets
  - Huge volumes of cross ISP traffic
  - ISPs use bandwidth throttling to limit bandwidth usage
  - Degrade QoS perceived at used end
- Sub-optimal uplink utilization
Comparison to P2P File Sharing

- **Similarities**
  - Distribution costs move from stream provider to network provider
  - Need incentives for end-users to contribute resources
  - Scalability needs uniform usage of link capacities (content replication proportional to popularity)

- **Differences**
  - QoS constraints essential for streaming
  - No time-consuming strategies against free-riding possible
  - Much higher churn due to huge fraction of short sessions
  - Much smaller number of shared items

Conclusion

- **P2P streaming efficient way to realize application level multicast**
  - Considers heterogeneous nodes
  - Conforms to IP network hourglass model
  - Self-optimizing

- **Widely used P2P application**
  - PPLive: 75 million global installed base and 20 million monthly active users (in 2007)