LiveShift
Current Progress and Challenges

Fabio Victora Hecht

Department of Informatics IFI, Communication Systems Group CSG, University of Zürich

Motivation and Objectives
Progress Report
Current and Future Work
Motivation

- Increasing IP connectivity
- Success of video sharing websites and applications
- High traffic demands, flash crowd effect
  - Infrastructure + network administration
- Drawbacks of current solutions
  - Market highly concentrated
  - Proprietary solutions
  - Censorship
  - On-demand/live segmentation
  - Low quality
  - High delay on live (SopCast can be more than 1 minute)
Motivation

Google Search Volume

Source: Google Trends
Proposed Solution

- **Peer-to-peer** streaming
  - **Scalable**, copes with flash crowds
  - Able to circumvent **censorship**
  - Low **cost** to distribute (cost is shared among peers)

- **Open** source, based on open standards
  - Community-based
  - No central control
  - **Robust**

- **Goals**
  - Maximize Quality of **Experience** for user
    - Delay, video quality, scalability
  - Minimize **Cost** for ISP, content provider
    - Overhead introduced by P2P
LiveShift

- Peer-to-peer video streaming application
- Combines Live streaming and Video-on-Demand
- Demo presented in P2P’08 in Aachen
  - Idea and base functionality implemented and presented
  - Feedback from participants taken into consideration
  - Work continued
- Recent changes
  - Mesh-pull distribution (was multiple-tree-pull)
  - Notion of larger segments composed of small blocks
LiveShift – Mesh

- LiveShift distributes video blocks via a mesh-pull approach
  - Video is divided in small blocks
  - Peers exchange blocks with no defined structure (e.g. tree)
    - Normally with a limited number of peers (neighbors)
  - Peers pull (request) each block
LiveShift – Mesh

- When compared to tree approaches

**Advantages**
- Flexible fanin and fanout
- Easy to use all available **bandwidth**
- Copes better with **churn**
  - Impact of peer leaving is lower
  - No structure maintenance to perform

**Disadvantages**
- Overhead: more control messages involved
  - One request per block
  - Normally involves exchanging of **bitmaps**
LiveShift – Block Size

- Tradeoff
  - Smaller block
    - Less delay
    - More requests
      - To DHT/tracker to find peers that have a block
    - Larger address space
      - More lookups
      - Larger identifier length
LiveShift – Blocks & Segments

- The compromise
  - Larger **segments** composed of smaller blocks
  - Segments
    - Identified by **channel + start time**
    - DHT associates segments with peers that have at least a block
    - Peers exchange **bitmaps** of which blocks they have in a segment (neighborhood)
  - Blocks
    - Identified by **segment + block#**
    - Requested **directly** between peers
    - Reply may contain the **video** data
    - In LiveShift, **time-addressed**
      - Size in bytes may vary
LiveShift – Neighbor Finding

- **Current** mesh-pull implementations
  - Step 1: **find** possible neighbors (normally central tracker)
  - Step 2: start **exchanging** bitmaps (done periodically)
  - Step 3: request/receive blocks

- **Result**
  - High delay for live streaming
  - High startup time for VoD
  - Low **QoE**

- **LiveShift approach**
  - Step 1: find neighbors (DHT)
  - Step 2: request/receive blocks (bitmaps go **piggybacked**)

- **Lower delay/startup time** is foreseen
LiveShift Architecture

- Encoder
- Player
- Controls
- GUI
- Assembler
- Player Sender
- Tuner
- Core
- Storage
- Scheduler
- Neighbor Finder
- Network
- Signaling/Video
- DHT
LiveShift – Publishing

- Encoder (JVLC)
  - Encodes video from any source using any codec
  - Outputs MPEG-TS in RTP

- Assembler
  - Receives video packets from Encoder
  - Distributes them into blocks
  - Distributes blocks into segments
  - Segments are announced on DHT

- Storage
  - Stores blocks and segments
LiveShift – Watching

- Tuner
  - Coordinates other components

- Neighbor Finder
  - Periodically checks DHT for other peers
  - Decides which and how many neighbors to keep

- Scheduler
  - Decides which blocks to request
  - Decides which neighbors to request
  - Reply may come with video block

- Player Sender
  - Verifies which blocks have arrived
  - Sends video to Player
  - Decides what to do if blocks are missing
LiveShift – Distributed Storage

- Peers participate in a **P2P storage** of video segments
- Storage in each peer is **limited**
- **Challenge**
  - **Ideal distribution** of stored segments
  - Very low or no communication **overhead**
  - Related work not found yet on this specific situation
LiveShift – Distributed Storage

- Current strategy
  - First, **fill** up the allowed storage space with every segment received
  - When full, selectively **delete** segments with a two-factor system: redundancy and time
    - **Redundancy**: probability P of keeping a segment
      \[ P = \text{(redundancy/swarmsize)} \]
    - **Time**: If not keeping it by the above probability, gives priority to newer segments

- A global challenge in P2P long-time storage
  - In order to have availability of blocks, it needs availability of **peers**
  - Situation improves with good **reputation** mechanisms or deployment of **superpeers** (e.g. owned by ISP, content provider)
LiveShift Reputation

- PSH
  - By Bocek et al.
  - Expected to help spread blocks faster
    - Higher success ratio
  - Question: does it help to give a benefit to share what is not being watched?
    - Consistent ID needed to keep long-term reputation score
      - Outdated information?
      - Space requirements?
    - Interest match enough to produce encounters?
Near Future Work

- Optimization
  - Policies for
    - Selecting neighbors
    - Localization of traffic
    - Scheduling blocks
      - Requesting several blocks with one request
      - Push vs. pull approach
    - Storing segments
  - Improve scalability
Near Future Work

- Evaluation
  - Run on EMANICS Lab
  - Study policies with two high-level objectives
    - Presented in introduction
  - ISP-assisted localization

- Paper showing LiveShift
  - Architecture
  - Parameter study
  - Evaluation
    - Compare to other implementations? How? Which?
Future Work

- Further improvements
  - PSH reputation mechanism

- Publish code as **open source** project
  - Only 2 projects found yesterday on Source Forge
    - One seems inactive – no releases or files – only one developer
    - One released last file in 2006 – also only one developer
  - Question on whether it is **too early**
  - Question on whether it will be **too late**
    - Pioneering is generally good
    - Easier to attract attention early
Preliminary Conclusions

- P2P Streaming is feasible
- Major challenges
- Vast opportunity for improvements
- Work is in early stage
- Software developed so far permits evaluation of different algorithms in a realistic environment
The End

Thank you!

hecht@ifi.uzh.ch