Distributed Systems
236351
Tutorial 9 - P2P Streaming
SplitStream,
CoolStreaming/DONet,
mTreebone

Yehontan Buchnik
(Based on Dolev Adas slides)

Technion

December 27, 2018
Motivation

- Over the years, streaming of video from the internet has become more and more popular
  - Over the years, streaming of video from the internet has became more and more popular
- Streaming can be done using a centralized solution
  - Youtube, Netflix, etc.
- However, the stream provider must incur the cost of keeping the video server (datacenter)
  - Most of these services are either paid or ad supported
- An alternative is the P2P solution, where the stream provider supports a low cost server and uses user donated bandwidth and space
Motivation

P2P Live Video Streaming

- A variation of P2P streaming where the streamed content is generated by a single source
  - TV channel, Live feed of an event, etc.
- Consumers are interested in viewing the content in a timely manner
- Some commercial P2PTV applications were developed in China
  - PPLive, PPStream
- Many more P2P live streaming systems were suggested in the literature
  - GridMedia, Coolstreaming/DONet, Splitstream, Prime, mTreebone and more
- P2P Live Streaming in not trivial
  - How to organize the clients?
  - How to stream the video?
  - How to deal with additions/departures (Churn)?
Motivation
First Attempt - A Tree-Based Layout

- Organize all clients in a single dissemination tree
- Who organizes the clients?
  - The source server (centralized organization)
  - Fully distributed algorithm
- A node can either an interior node or a leaf node - but never both
  - A leaf node donate nothing to the bandwidth
  - A failure of an interior node disconnect a whole sub-tree
SplitStream - 2003
Microsoft Research
SplitStream

Overview

- Split the stream into $k$ stripes
- Each stripe is broadcast over its own tree
- Forms a *forest* of such stripe trees
  - Each node should be an interior node in one tree and a leaf node in all others
  - A node denotes the same bandwidth it receives
  - Different nodes can configure different constraints in their in/out bandwidth
- Resilient to $k$ failures
- Can correct the date if some stripes have failed
  - Multiple Description Coding (MDC) - loss of one or more split streams will result in a loss of video quality
  - Erasure coding - introduce one or more redundant split streams
SplitStream
Overview
SplitStream

Challenges

- Building the forest
- Limiting node degree
- Joining the network
SplitStream
Building The Forest

SplitStream uses **Scribe** which is an application-level communication system built upon **Pastry**. Pastry is a **Chord** style DHT.

- Scribe can form a tree over Pastry named **group**
  - A pseudo-random key known as **groupId** is generated
  - Each node searches for the groupId’s root
  - The tree is formed by the union of the routing tables of the group’s members
  - Messages are multicast from the root using reverse path forwarding
SplitStream
Building The Forest

In Pastry (and Chord) each node try to route a request to the node with the longest common prefix with the required object

- Based on Pastry routing method, if every stripe is assigned with a groupId that is different by the most significant digit from other stripes, then every node has an equal chance of becoming an interior node in some tree. Thus the forwarding node is approximately balanced.

- Full details can be found in the paper.
SplitStream
Building The Forest

Source

Stripeld 0x  Stripeld 1x  Stripeld Fx

- Nodelds starting 0x
- Nodelds starting 1x
- Nodelds starting Fx
- Nodelds starting 2x..Ex
SplitStream
Limiting Node Degree

- When a node that has reached its maximal out-degree receives a request from a prospective child, it has to abandon one of its other children.
SplitStream
Joining The Network

- When a node $q$ wishes to join the network it sends a request message to a known node $p$.
- $p$ acts as followed:
  - $p$ adopts $q$
  - If $p$ out-degree has reached to its maximal value, $p$ abandons one of its children following the next policy:
    1. $p$ searches for children whose their stripeId does not share a prefix with the node's id - if the prospective child is among them, $p$ abandons it, else it picks a random node among those group and abandons it.
    2. If no such child is exists, $p$ is an interior node in a single stripe tree. Thus, $p$ searches for the children whose has the shortest prefix match with that stripeId - if the prospective child is among them, $p$ abandons it, else it picks a random node among those group and abandons it.
- The abandoned child, asks from one of its former siblings to adopt it and repeat the above.
- If no one of the family can adopt the orphan node, it register to the Spare Capacity Group.
SplitStream
Joining The Network

(1) 001* 0800 0800 0800 0800
     089* 08B* 081* 9*

(2) 001* 0800 0800 0800 0800
     085* 08B* 081* 001*

(3) 0800 0800 0800 0800 0800
     089* 08B* 081* 001*

(4) 0800 0800 0800 0800 0800
     089* 08B* 081* 001*
SplitStream
Spare Capacity Group

- Another Scribe group which consist of nodes that have less children than their maximal out-degree
- Each node report the stripes in whom it is an interior node and its capacity
- The orphan node asks to join the group and is directed to the first node that can adopt it (using DFS)
CoolStreaming/DONet - 2005
CoolStreaming/DONet

Overview

- A mesh-pull based streaming architecture
- CoolStreaming constructs a random neighboring overlay using **SCAMP**
  - SCAMP is a Scalable Gossip Membership protocol
  - Distributes membership messages among nodes
  - Scalable, light-weight, and maintains uniform partial view at each node
- A peer randomly selects a fixed number of peers to connect with
- Each video is split into fixed length segments
- A node is either receiver, supplier, or both, depending on the availability of its segments
CoolStreaming/DONet

Buffer map

- Every node generates periodic messages letting others know that it is still alive and what segments it holds.
- The buffer map message **BM** is used to indicate what segments are stored in this node’s buffer.
- Exchanged all the time with the neighbours, and used to scheduled which parts to be downloaded from where.
The scheduler determines what segment to be downloaded when and from where

- Should meet the **deadline** for the segment and not exceed the **offered bandwidth**
- Starts by scheduling the segment with the fewest suppliers, since it assumes this will have most difficulty meeting the deadline
- If a segment is available at more than one neighbor - the neighbor with the highest bandwidth and from which the estimated segment arrival time is before the deadline is selected
mTreebone

Overview

- A hybrid mesh/tree overlay for application-layer live video multicast
- Studies have found that, nodes with higher age tend to stay longer
- The key idea is to identify a set of stable nodes to construct a tree-based backbone, called treebone, with most of the data being pushed on the backbone
- In addition, the whole network is further organized through an auxiliary mesh overlay to fully exploit the available bandwidth between overlay nodes
  - Similar to CoolStreaming
mTreebone

Overview
mTreebone
Overview
mTreebone

Departure Of a Node

- When an unstable node, such as node A fails or leaves, it will not affect the data pushed along the treebone.
- If a stable node leaves, the impact can be remarkably mitigated with the help from the mesh overlay.
mTreebone

Treebone optimization

While running a node may discover that it can optimize the tree in the sense of latency. For these mTreebone implements two operations

- High-degree preemption and
- Low-delay jump
Treebone optimization

High-degree preemption

Each treebone node $x$ periodically checks whether it has more children than a node that is closer to the source in the treebone. Such a node, referred to as $y$, could either be the parent of $x$ in the treebone, or a node known from $x$’s local node list. If so, node $x$ performs the High-degree preemption in which node $x$ will then preempt $y$’s position in the treebone, and $y$ will rejoin the treebone. In practice, $y$ can simply attach itself to $x$. 
Treebone optimization

Low-delay jump

Each treebone node $x$ periodically checks whether there are nodes closer to the source than its parent. If so and one such node, say $y$, has enough available bandwidth to support a new child, node $x$ will leave its original parent and attach itself to $y$ as a child.
For More Details

- SplitStream
- CoolStreaming/DONet
- mTreebone
- BitTorrent Live
  - BitTorrent Live
  - Lecture