P2P Live Streaming

Fatemeh Rahimian
fatemeh@sics.se

Amir Payberah
amir@sics.se
Outline

• Introduction
• P2P multicast algorithms
• Comparison
• Future work
• Summary
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
- Single tree approaches
- Improved single tree approaches
- Mesh-based approaches
- Multiple tree approaches
- Mixed approaches
- Comparison
- Future work
- Summary
Introduction

• P2P multicast algorithms:
  ▪ Infrastructure-based (two-tier) approaches
  ▪ Single tree approaches
  ▪ Improved single tree approaches
  ▪ Mesh-based approaches
  ▪ Multiple tree approaches
  ▪ Mixed approaches
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
  - Mesh-based approaches
  - Single tree approaches
  - Improved single tree approaches
  - Multiple tree approaches
  - Mixed approaches
- Comparison
- Summary
Infrastructure-based Approaches

- **Two separate sections:**
  - The actual p2p network
    - Core of system
    - Generally resourceful nodes
  - The outer network
    - End users
    - Doesn't take active part in distribution of information
Related Algorithms

• Overcast [1]
• Scattercast [2]
Overcast
Overcast

• Main parts of overcast:
  ▪ Central source
  ▪ Internal Overcast nodes (pool of nodes)
  ▪ Standard HTTP clients

• Tree of internal nodes rooted at the source

• Maximize the available bandwidth from the source to all nodes
Overcast (Join internal nodes)

- At booting up, the node contacts a global, well-known registry
- Registry provides a list of Overcast networks to join.
- Initially chooses the root as its parent.
  - Calls it current parent
Overcast (Join internal nodes)

- Begins a series of rounds
  - Attempts to locate itself further away from the root
  - Without sacrificing bandwidth back to the root

- In each round
  - Considers its bandwidth to current parent
  - Considers bandwidth to current parent through each of current parent’s children
  - If the bandwidth through any of the children is about as high as the direct bandwidth to current parent

- A node periodically re-evaluates its position in the tree.
Overcast (Leave/Failure internal nodes)

• When a node detects that its parent is unreachable:
  ▪ It will relocate beneath its grandparent, if not ...
  ▪ Continues to move up its ancestry until it finds a live node.

• Nodes maintain an ancestor list
  ▪ Avoid cycles
Overcast (Join client nodes)

- Joining a group consists of:
  - Selecting the best server
  - Redirecting the client to that server

- The client issues an **HTTP GET** request with a URL to group.
  - The host name of URL is the name of root.

- The root decides where to connect the client to the multicast tree.
Scattercast
Scattercast

• The same as Overcast, but ...

• The nodes in core network adapt dynamically their connectivity to the client load
Advantages & Drawbacks

• Advantages
  ▪ Transparent to the user
    • They can use standard client applications
  ▪ Total control over network traffic by owner
  ▪ The problem of fairness, security and dishonest nodes are marginal

• Drawbacks
  ▪ Vulnerable to DoS attacks
    • The number of core hosts is finite and their location may be known
  ▪ Doesn't cope very well with flash crowd and sudden increase of traffic
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
- Single tree approaches
  - Improved single tree approaches
  - Mesh-based approaches
  - Multiple tree approaches
  - Mixed approaches
- Comparison
- Future work
- Summary
Single Tree Approaches
Related Algorithms

- PeerCast [5]
- Scribe [6]
- NICE [7]
Peercast
PeerCast

• PeerCast is today a fully deployed application
  ▪ Mostly used for independent, small-scale Internet radio broadcasts

• Organizing the group members into a self-organized, source-specific, spanning tree.
PeerCast (Join)

- The node contacts the source of the stream
  - Each live stream has a unique URL
  - The source information is embedded in it

- If source is unsaturated
  - Accepts a data transfer session setup request.

- If not,
  - the request is send to source immediate children.
    - Which child:
      - Random
      - Round-Robin
      - Smart placement

- The process continues iteratively,

- If node is unable to find an unsaturated node sends unavailable error to the upper application-layer
PeerCast (Leave/Failure)

- On leave, it forwards a valid target $t$ to its descendants.

- Each node is aware of two nodes:
  - Its parent
  - Source

- After leaving
  - Each descendant tries to recover by contacting targets
  - Or only children of unsubscribed node attempt to recover by contacting target.

- In failure, only source is identified as target.
Scribe
Scribe

- Scribe is a scalable application-level multicast infrastructure

- It is built on top of Pastry

- A Scribe node
  - May create a group
  - May join a group
  - May be the root of a multicast tree
  - May act as a multicast source
Scribe (Creating group)

• Each group has a unique group-Id.
  ▪ Hash of the group’s textual name

• Sends CREATE message
  ▪ With group-id

• Pastry delivers this message to the node with closest numerical node-Id: rendezvous point.

• The rendezvous point is the root of tree for the group
Scribe (Join)

- Send JOIN message
  - With group-id

- Pastry routes to rendezvous point
  - If intermediate node is forwarder
    - Adds the node as its child
  - If intermediate node is not a forwarder
    - Creates child table for the group, and adds the node
    - Sends a JOIN towards the rendezvous point
      - becomes forwarder
  - Terminates JOIN message from the child.
Scribe (Leave/Failure)

• If the node has no children in its table, it sends a LEAVE message to its parent
  ▪ The message travels recursively up the multicast tree
  ▪ The message stops at a node which has children after removing the departing node
Scribe (Leave/Failure)

- Non-leaf nodes send heartbeat message to children
  - Multicast messages serve as implicit heartbeat
- If child does not receive heartbeat message
  - Assumes that the parent has failed
  - Sends a JOIN message to the group-id

- If rendezvous point fails
  - The state associated with a rendezvous point is replicated across k closest nodes
  - The children detect the failure and send a JOIN message which gets routed to a new node-id numerically closest to the group-id
Scribe (Data Delivery)

• Source sends MULTICAST message to the rendezvous point

• Source caches the IP address of the rendezvous point
  ▪ So that it does not need Pastry for subsequent messages
NICE
NICE

• Support large receiver sets with small control overhead

• Hierarchical membership
  ▪ Clients are assigned to different layers
  ▪ Each layer is partitioned into a set of clusters
    • The size between $k$ and $3k - 1$ ($k$ is a constant).
  ▪ All hosts belong to the lowest layer $L_0$
  ▪ One host selected as leader
  ▪ Leaders of clusters of $L_i$ join layer $L_{i+1}$
NICE (Join)

• Rendezvous Point (RP)
  ▪ All hosts know the RP host

• Join procedure
  ▪ Contact RP to get the cluster members of the highest layer
  ▪ Loop until reach layer 0
    • Query the members of the returned cluster and find the closest one, X
    • Get the members of the child-cluster of X
NICE (Leave/Failure)

- On leave
  - Send a leave message to all clusters it belongs

- On failure
  - Other hosts detect the leave by not receiving the periodic refresh of $H$

- If $H$ is leader
  - Each remaining member, $J$, select a new leader independently
  - Multiple leaders are resolved by the exchange of refreshes
NICE (Maintenance)

• Cluster-leader periodically checks the size of its cluster in layer $L_i$
  ▪ If the cluster size exceeds the $3k - 1$ limit
    • Split the cluster into two equal-sized clusters
  ▪ If the cluster size is under $k$
    • The leader finds a closest host in layer $L_{i+1}$ and merge with it

• Each member, $H$, in any layer $L_i$ periodically probes all members in its super-cluster, to identify the closest member
  ▪ If a host, $J$ is found, then H joins to the cluster under the $J$
NICE (Data Delivery)

• Control paths
  - Exchange periodic state refreshes
  - For a host $X$, the peers on its control topology are the other members of the clusters to which $X$ belongs

• Data paths

Procedure: $\text{MulticastDataForward}(h, p)$

$$\{ h \in \text{layers } L_0, \ldots, L_i \text{ in clusters } Cl_0(h), \ldots, Cl_i(h) \}$$

for $j$ in $[0, \ldots, i]$
  if ($p \notin Cl_j(h)$)
    $\text{ForwardDataToSet}(Cl_j(h) - \{ h \})$
  end if
end for
Advantages & Drawbacks

• Advantages
  ▪ Optimal with respect to transmission delay

• Drawbacks
  ▪ Doesn't share the load in even way
  ▪ Reacts badly to node failure
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
  - Single tree approaches
  - Improved single tree approaches
    - Mesh-based approaches
    - Multiple tree approaches
    - Mixed approaches
- Comparison
- Future work
- Summary
Related Algorithms

- ZigZag [8]
- BulkTree [9]
ZigZag

- It organizes the receivers into bounded size clusters and makes a multicast tree based on them

- Two important entities:
  - Administrative organization
    - Logical relation among peers
  - Multicast tree
    - Physical relation among peers
ZigZag (Rules)

• When a peer is not at its highest level can not have link to other peers.
• When a peer is at its highest level, can only link to its foreign subordinate.
• The members of a cluster at any layer get the content from their foreign head.
• The peers in each cluster periodically sends some control messages to its clustermate, its parent and its children
  ▪ Reachable
  ▪ Addable
  ▪ ...
ZigZag (Join)

1. If $X$ is a leaf
2. Add $P$ to the only cluster of $X$
3. Make $P$ a new child of the parent of $X$
4. Else
5. If $Addable(X)$
6. Select a child $Y$:
   \[ Addable(Y) \text{ and } D(Y) + d(Y, P) \text{ is min} \]
7. Forward the join request to $Y$
8. Else
9. Select a child $Y$:
   \[ Reachable(Y) \text{ and } D(Y) + d(Y, P) \text{ is min} \]
10. Forward the join request to $Y$
ZigZag (Leave/Failure)

(a) Before Failure

(b) After Recovery

Links to non-head members of cluster
A single link to a child
BulkTree

• Main idea:
  ▪ Organizing a set of close nodes (weak-nodes) into a strong super node
  ▪ Construct a tree structure over super nodes

• The size of each super node is \([k, 3k-1]\) and size of each leaf super node is \([1, 3k - 1]\).

• Each super node has a leader and a backup leader

• The leader nodes collect information about parent super node, other weak nodes in the same super node and their children super nodes
• The other weak nodes only collect information about parent super node and children super nodes.
BulkTree (Join)

- First contacts to the server and the server redirects it to its children

1. Select a node $X$ from $Leader(S)$: $D(H, X)$ is min;
2. If $Super(X)$ is a leaf
3. Add $H$ to the $Super(X)$;
4. Else
5. If $Addable(X)$
6. Add $H$ to the $Super(X)$;
7. Else if another node $Y$ in $Leader(S)$ is addable
8. Add $H$ to the $Super(Y)$;
9. Else send $Leader(X)$ to $H$ for contact;

- If after joining the new node, the size of super node is exceeded, it should be split and balanced
BulkTree (Leave/Failure)

- If H is the leader of super node, the backup leader becomes leader.

- If after departing a node the size of super node becomes lower than k, merging should be taken.
BulkTree (Data Delivery)

- First scheduled inside super node, and if needed ask its parent super node

- If the parent has the data, the leader chooses $k$ good nodes and these $k$ nodes then send $1/k$ data to receivers
Advantages & Drawbacks

• Advantages
  ▪ Optimal with respect to transmission delay

• Drawbacks
  ▪ Doesn’t share the load in even way
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
- Single tree approaches
- Improved single tree approaches
- Mesh-based approaches
- Multiple tree approaches
- Mixed approaches
- Comparison
- Future work
- Summary
Mesh-based Approaches
Related Algorithms

- Narada [3]
- Yoid [4]
Narada
Narada

- Narada is a multiple-source multicast overlay infrastructure

- It uses
  - Mesh as a control infrastructure
  - Tree as data delivery infrastructure

- The tree is constructed in a two-step process:
  - It constructs a mesh with desirable performance properties
    - Path Quality
      - Application interested metric: delay, bandwidth, ...
    - Limiting the number of neighbours
      - Controls the overhead of running routing algorithms
  - It constructs spanning trees of the mesh,
    - Each tree rooted at the corresponding source
Narada (Join)

- Each node gets a list of group members
  - By an out-of-band bootstrap
  - Does not need to be complete or accurate
  - Must contain at least one currently active group member

- Selects randomly a few group members from the list

- Sends requesting message to them to be added as a neighbour.
  - It repeats until it gets a response
Narada (Leave/Failure)

- On leave, it notifies its neighbours,
  - Propagated to the rest of the group members along the mesh.

- The leaving member continues forwarding packets for some time
  - To minimize transient packet loss

- Failures should be detected locally
  - By not receiving refresh messages from some node for a while
  - Propagate to the rest of the group

- Nodes are capable of detecting and repairing partitions.
Narada (Maintenance)

• Allows incremental improvement of mesh quality
  ▪ By adding and dropping of overlay links

• Members probe each other at random
  ▪ New links may be added depending on the perceived gain in utility in doing so.

• Members continuously monitor the utility of existing links,
  ▪ Drop links perceived as not useful
Yoid
Yoid

- Each member acts relatively independently

- It uses:
  - Tree
    - Efficiency
    - Multicast of application content
  - Mesh
    - Robustness
    - Broadcast of control and application content

- Each group has a groupId
  - yoid://rendezvous.name:port/groupName
Yoid (Rendezvous host)

• It is not part of tree-mesh

• Primary purpose of the rendezvous host is bootstrapping members.
  ▪ By informing each member of several current member, and optionally various other information about the tree.

• Each node talk to rendezvous in several cases:
  ▪ When joining
  ▪ When leaving
  ▪ Sends ping message to it (I'm alive)
  ▪ Informs rendezvous when the node becomes root of tree
Yoid (Mesh construction)

- Each member maintains a small number of neighbours.

- To insure a non-partitioned mesh topology:
  - Each member M establishes a small number of other members
    - Three or four
  - Selected randomly
  - Must not include members that are tree neighbours
  - Must not include members that have already established a mesh link to this member M
Yoid (Tree construction)

- A member may receive and transmit frames via:
  - Unicast IP
  - Scoped IP multicast
    - One hop
- Where multicast IP is used, a set of members are grouped as a cluster.
  - One member of the cluster is elected the head,
    - Is responsible for establishing a (unicast IP) parent neighbour
  - The other cluster members are called feet
    - Transmit and receive to/from the tree via the head
- Each member that cannot join a cluster or is the head of a cluster:
  - Is responsible for either finding a parent in the tree
  - Or deciding that no other member can be a parent
    - Becoming the root of the tree
- No loop prevention, but detection using Root Path
Yoid (Tree construction)
Advantages & Drawbacks

• Advantages
  ▪ A true p2p network
    • All nodes have the same role regardless of their placement, capabilities and resources
  ▪ Doesn't have single point of failure
  ▪ Resilient to massive host crashes and disconnects
  ▪ Support both single-source and concurrent multiple-source

• Drawbacks
  ▪ Complexity of management
  ▪ Steady flow of control messages between all nodes
    • limited size
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
- Single tree approaches
- Improved single tree approaches
- Mesh-based approaches
- Multiple tree approaches
- Mixed approaches
- Comparison
- Future work
- Summary
Multiple Tree Approaches
Related Algorithms

- Zebra [10]
- CoopNet [11]
- SplitStream [12]
- Orchard [13]
Zebra
Zebra

- Serves high quality live media to up to 100 clients.

- Two trees
  - A serving node in one tree should be leaf in other tree.

- Two parts of system:
  - Server proxy
    - Divides the media into two stripes
    - Maintains full system state
  - Client proxy
    - Update the server proxy on occurring events
    - Forwards data to its children
    - Sends data to its media player
Zebra (Join)

- New node communicate with server proxy.

- Server proxy determine which stripe should be served by new node.
  - Based on number of node serve each stripe.

- Server proxy return up to 10 node:
  - Available nodes
  - Splice-able nodes
Zebra (Leave/Failure)

- Immediate children of disconnected node keep their subtree
- Inform the server proxy and they try to reconnect to system
CoopNet
CoopNet

- CoopNet complements the client-server framework rather than replaces it.
  - There is still a directly connection between servers and clients
  - CoopNet is only invoked when the server is unable to handle the load imposed by clients.

- Uses MDC
CoopNet (Tree Management)

- Goals in constructing and maintaining trees:
  - Short and wide tree
  - Efficiency versus tree diversity
    - Diversity: minimizes chance of disruption
    - Efficiency: matches underlying network topology
  - Quick join and leave
  - Scalability
CoopNet (Join)

- The new node contacts the server
  - Informs its available network bandwidth

- The server responds with a list of designated parent nodes, one per distribution tree
  - Using a top-down approach until find nodes with spare capacity
  - Select randomly between them

- Upon receiving the server’s message, the new node sends (concurrent) messages to the designated parent nodes
  - To get linked up as a child in each distribution tree.
CoopNet (Leave/Failure)

- On leave the departing node informs the server
  - The server identifies the children of the departing node
  - Executes a join operation on each child

- Each node monitors the packet loss rate of each distribution tree

- If the packet loss rate reaches an unacceptable level
  - The node contacts its parent to check if the parent is experiencing the same problem.
  - If so, the source of the problem is upstream of the parent and the node leaves it to the parent to deal with it.
    - The node also sets a sufficiently long timer
  - If the parent is not experiencing a problem or it does not respond, the affected node will contact the server and execute a fresh join operation for it
SplitStream
SplitStream

- It is implemented based on Pastry and Scribe.
- Pastry key is used as groupId.
- The union of routes from the group members toward each groupId form the group multicast tree.
- The content is split into k stripes.
- Using a separate tree to multicast each of them.
- A node is an interior node in at most one stripe tree and is a leaf node in all the other ones.
  - interior-node-disjoint.
SplitStream (Join – first step)

- Attempting to join the stripe tree directly
SplitStream (Join – second step)

• If first step fails, it looks for a parent in the spare capacity
  ▪ anycast
Orchard
Orchard

• Orchard is an algorithm for ALM of video streams over unstructured P2P systems.

• Each node maintains a neighbour set.

• No peer forwards more descriptions than it receives.

• A peer does not need to know the source or any other specific peer.

• Splitting up into several substreams using MDC.

• Building a forest of separate spanning trees, each tree serving a single substream.
Orchard (Join)

- Join at source

- Exchange Descriptions

- Redirection

- Redirection through coloured nodes (temporary)
Orchard (Leave/Failure)

- Any deal with departed/failed node will be cancelled
  - Also redirect deals

- Find and use backup parent

- Rejoin
Orchard (Maintenance)

- Changing colour
- Changing parent

[Logo]
Advantages & Drawbacks

• Advantages
  ▪ Shares the load
  ▪ Uses the resources

• Drawbacks
  ▪ Complex management
Outline

• Introduction
• Infrastructure-based (two-tier) approaches
• Single tree approaches
• Improved single tree approaches
• Mesh-based approaches
• Multiple tree approaches
• Mixed approaches
• Comparison
• Future work
• Summary
Related Algorithms

- Bullet [14]
- PULSE [15]
Bullet
Bullet

• Aimed at maximizing the bandwidth delivered to the receivers through download of disjoint data from multiple peers

• Uses MDC to make data recovery more efficient

• By building a tree, it purposefully disseminate disjoint objects to different clients

• Nodes are responsible for locating peers that hold missing data objects (Using RanSub Protocol)
• RanSub Protocol
  ▪ Collect message
    • start at the leaves and propagate up the tree, leaving state at each node along the path to the root
  ▪ Distribute message
    • start at the root and travel down the tree, using the information left at the nodes during the previous collect round to distribute uniformly random subsets to all participants
Bullet

• Informed Content Delivery Techniques
  ▪ Messages contain summary tickets of the objects available at a subset of the nodes in the system
  ▪ Nodes uses BloomFilter to perform approximate fine-grain reconciliation

• Request data objects from remote nodes that have significant divergence in object membership
PULSE
PULSE

• Some concepts:
  ▪ Lag
    • The age of chunk with respect to the current media clock.
  ▪ Sliding window
    • Is used to output a stream of chunks with desired max. loss ratio
  ▪ Zone of interest
    • Collects the chunks which will be needed soon
PULSE (Cont'd)

- Main components of each peer:
  - Data buffer
    - Used to collect and store chunks before playing
  - Knowledge record
    - The information about remote peers
  - Trading logic
    - Determines which chunk should be requested neighbours and choose and schedule the chunks that are to be sent

- Two groups of neighbours:
  - MISSING
    - Having overlapped trading window
  - FORWARD
    - No overlap
PULSE (Algorithm)

• Main parts of algorithm:
  ▪ Peer selection
    • BitTorrent tit-for-tat idea
  ▪ Sending chunk selection
    • Latest send first, random
  ▪ Requesting chunk selection
    • Rarest chunk across the neighbourhood
Outline

• Introduction
• Infrastructure-based (two-tier) approaches
• Single tree approaches
• Improved single tree approaches
• Mesh-based approaches
• Multiple tree approaches
• Mixed approaches
• Comparison
• Future work
• Summary
Comparison

• Single tree structures
  ▪ Optimal with respect to the transmission delay
  ▪ Don’t share the load in an even way among the participating peers
  ▪ React quite badly to node failures

• Multiple tree structures
  ▪ Good solution
  ▪ Requires a heavy control traffic to manage many trees

• Mesh structures
  ▪ Very robust
  ▪ Adapt well to variations in the network conditions
    ▪ By continuously monitoring multiple paths
  ▪ Costly to maintain
  ▪ Don’t scale well for large groups of users

• Mixed structures
  ▪ Robustness and performance advantages of the mesh
  ▪ Smooth management of the single tree
## Comparison

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Target Audience</th>
<th>Control Overhead</th>
<th>Response to Load</th>
<th>Response to Transience</th>
<th>Management Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-tier</td>
<td>large (bounded)</td>
<td>low</td>
<td>w/threshold</td>
<td>good</td>
<td>low</td>
</tr>
<tr>
<td>Mesh</td>
<td>small/medium</td>
<td>high</td>
<td>good</td>
<td>good</td>
<td>average</td>
</tr>
<tr>
<td>Single Tree</td>
<td>any</td>
<td>low/medium</td>
<td>good</td>
<td>bad</td>
<td>variable</td>
</tr>
<tr>
<td>Multiple Tree</td>
<td>medium/large</td>
<td>medium</td>
<td>good</td>
<td>quite good</td>
<td>high</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Infrastructure-based (two-tier) approaches
• Single tree approaches
• Improved single tree approaches
• Mesh-based approaches
• Multiple tree approaches
• Mixed approaches
• Comparison
• Future work
• Summary
Future work - ForestCast

• Main goals
  ▪ Maximize utility
  ▪ Minimize latency

• Nodes can have three roles:
  ▪ source – the node which has the video to be streamed
  ▪ server – central server that constructs the trees
  ▪ peer – a node (customer) which downloads and/or uploads the stream

• The server will have complete information about every peer.
  ▪ New nodes will provide server with their own bandwidth capacity.
  ▪ The server will also approximate the latency of the peer
  ▪ The server will inductively construct trees, which it maintains as peers join, leave, and fail.
ForestCast

- Open Slots
- Total Latency
- State of the existing trees

A set of parameters

LS, CP or ... using different heuristics

Decisions
Outline

- Introduction
- Infrastructure-based (two-tier) approaches
- Single tree approaches
- Improved single tree approaches
- Mesh-based approaches
- Multiple tree approaches
- Mixed approaches
- Comparison
- Future work
- Summary
Summary

• Infrastructure-based (two-tier) approaches
• Single tree approaches
• Improved single tree approaches
• Mesh-based approaches
• Multiple tree approaches
• Mixed approaches
References


References


References


Questions?
&
Comments!