Resilient Tree-based Live Streaming for Mobile Scenarios

Benjamin Schiller and Giang Nguyen and Thorsten Strufe
P2P Networks
TU Darmstadt
[schiller, nguyen, strufe]@cs.tu-darmstadt.de

Abstract—Using Peer-to-Peer technology to deliver live video streams to mobile devices is a promising approach. It allows service providers to scale their video distribution without increasing their cost. As the resources are replicated at the edge of the network, mobile devices can take advantage of close-by peers in order to get the required data faster. This, however, is challenging due to the highly dynamic nature of the participating mobile devices. Hence, the video distribution overlay needs to adapt quickly to changes in the available bandwidth as well as the location of peers. Also, it should be resilient to arbitrary disconnects as well as targeted attacks. In this paper, we introduce a multi-tree-push streaming system which takes the upload capacities of mobile devices into account and arranges the overlay connections based on their position in the network topology. Our demonstrations show that our system is resilient to churn and attacks while running on desktop machines and mobile devices.

I. INTRODUCTION

Mobile devices such as smart phones and tablets have become omnipresent in the past few years. They are now equipped with powerful hardware and mature operating systems. This opens up opportunities for new services and new ways to deliver traditional services such as IPTV to mobile devices. Given the popularity of such devices and services, the provision of the required resources for a service of this scale is challenging.

Peer-to-Peer (P2P) streaming has been a scalable and cost-efficient solution to deliver live video streams by using the upload capacities of participating peers. However, it is not trivial to build an efficient and robust streaming system when dealing with unreliable end-hosts with quickly changing network connections. There are three main approaches of P2P-based live streaming systems: tree-based, mesh-based and hybrid.

Based on publish-subscribe concepts, tree-based systems have low latency which is a clear advantage for delivering live content. In single-tree systems, the failure of a small number of peers can have a big impact on a large part of the network which makes these systems the least resilient choice. In a clear contrast, peers in a mesh-based system form a mesh topology, similar to BitTorrent swarms which are more resilient to failures and churn due to path redundancy. However, requesting every video packet introduces higher delays and additional overhead. Hybrid systems combine the advantages of tree-based and mesh-based systems, but it is unknown how they resist against Denial-of-Service (DoS) attacks. A multi-tree system splits the video stream into several logical parts called stripes and delivers them using separate dissemination trees. This leverages the low latency of tree-based approaches while decreasing inter-peer dependencies. Moreover, it has been proven that multi-tree systems can be resilient against DoS attacks [1]. Nonetheless, how the system performs in real world conditions, especially in mobile scenarios, is not yet known.

In this work we demonstrate a deployable multi-tree-based system for P2P-based live video streaming. It runs on both desktop PCs and Android-based hand-held devices, including mobile phones and tablets. Additionally, it provides controlling and monitoring functionalities which help during the development phase and visualize the topology during a demonstration. Besides, the generic architecture of
the system is ready to harbor the implementation of other streaming system classes.

II. STREAMING SYSTEM

We developed a multi-tree-push streaming system which is resilient to attacks as well as the network dynamics of mobile devices. The video stream is segmented into several stripes. Video packets in each stripe are distributed over a separate spanning tree. The trees are constructed in inner-node disjoint and span all peers respectively. Being disconnected in one tree in this case does not mean losing the whole video stream. Negative impacts of packet loss can be further reduced by using forward error correction or special video codecs.

Every peer attempts to optimize the overlay by observing and adapting its local neighborhood. This only requires information about a peer’s neighbors and changes only affect their connections. Therefore, a single attacker can not collect information about the whole network as a peer following the regular protocol.

Each peer optimizes its own set of neighbors (including parent and child nodes) whenever the available upload bandwidth differs from their current usage or their neighbor list changes. In case a peer has more children than it can serve, it pushes a child down the tree, i.e., it asks one of its other children to overtake the parent role (cf. Figure 1). On the contrary, a peer who has enough capacities available to upload to another child pulls up another peer, i.e., it asks a child to hand-over one of its children (cf. Figure 2).

We implemented four strategies to decide which peer to pull up or push down: random, bandwidth, location, and location/bandwidth. As a baseline, peers using the random strategy select children to push down or pull up randomly. In the bandwidth strategy, peers with high bandwidth are placed closer to the source. Since mobile devices have rather limited upload bandwidths, this strategy automatically places mobile peers at the bottom of the trees and prevents them from being overloaded by the responsibility to upload too much data. Using the location-based approach, topologically close peers are clustered based on an approximation of their network position. This is especially beneficial for mobile peers as it automatically connects them to closeby peers, e.g., nodes in the same WiFi network instead of far-away peers. Lastly, the location/bandwidth-strategy combines information about bandwidth and location of neighbors to decide which peer to pull up or push down.

The streaming system consists of three core components: tracker, source, and peer (cf. Figure 3). The tracker maintains a list of all peers currently connected to the respective overlay. It assists peers in the bootstrapping stage by providing a list of other peers to contact. A source is the initial provider of a live video stream. It is the root of all trees used for the push-based distribution of the data in each stripe. A peer is any client that wants to receive a specific video stream. After obtaining some bootstrap nodes from the tracker, it joins the overlay, becomes part of each stripe’s tree topology, and participates in the distribution of the video stream.

For demonstration purposes, we added the monitor component which allows for the surveillance of
Fig. 4. Monitor component displaying the topology of the system and properties of selected peers and sources.

a running system (cf. Figure 4). It collects topology information and runtime statistics from sources and peers. Thereby, it facilitates live monitoring of the system’s topology and the properties of sources and peers. In addition, the monitor is capable of changing the configuration of peers and sources, prompt them to leave the system, or even shut them down immediately to simulate attacks on parts of the network.

III. DESIGN AND IMPLEMENTATION

The architecture of the streaming system has been developed in analogy to our simulation model for P2P-based streaming systems [2]. The peer and source components both provide a streaming layer supplying all parts required to distribute the video stream between the entities. The packet generator creates packets from an input video stream. The buffer stores incoming data that is forwarded to the video player. The task of the connection maintenance is to build the streaming overlay and keep it connected while the topology optimization performs optimizations of the overlay. Both directly interact with the bootstrap part of the discovery services that allows peers to discover others by querying the membership management of the tracker.

All components of our architecture can be replaced by other implementations. This allows for the evaluation of different optimization and buffering strategies and even different overlay topologies.

Currently, the streaming and measurement systems are implemented in Java. All components can be run on desktop machines (Windows, Mac, Linux). Source and peer are also available for Android devices. The packet generator creates an RTP stream from an input video stream which can easily be distributed among any number of stripes. On all platforms, VLC is used to display the received video streams and generate the RTP packets. The code is open source and licensed under the GPL.

IV. PLANNED DEMONSTRATION

We plan to show the following features:

**Streaming system:** Demonstrating the multi-tree-push live streaming system which works on desktop machines (Windows, Mac, Linux) and mobile devices (Android) (cf. Figure 5).

**Optimization strategies:** Impact of different optimization strategies on the overlay topologies as well as performance measures.

**Monitoring system:** Visualizing the topology of the running streaming system and modify the behavior of nodes (cf. Figure 4).

**Attack resilience:** Showcasing different scenarios, including attacks on the participating nodes, to demonstrate the system’s resilience.

Fig. 5. Two peers running on Android devices, a Mac running sources and peers, and a Linux machine hosting multiple sources.

The audience will be able to download the peer application to their Android devices, select an available stream, and join the desired overlay.

REFERENCES
