Internet of Things / Peer-Project Proposal

Group: Alfa - Revision: 1

Emil 'Skeen' Madsen
Aarhus University
Aarhus, Denmark
skeen@cs.au.dk

Rasmus Mosbech Jensen
Aarhus University
Aarhus, Denmark
20105109@cs.au.dk

ABSTRACT

We purpose a modern peer-2-peer content distribution network, based around the somewhat unoriginal idea of distributing music files.

Several systems alike what we purpose have previously been implemented, with one of the first being Napster by Shawn Fanning. Napster utilized a centralized server to index the files of the network, and peer-2-peer sharing via. direct downloading.

Most popular music distribution networks utilize a far more traditional client- server architecture, examples are; Spotify, Apple Music, SoundCloud and Tidal.

We propose a fully decentralized architecture, which ensures high scalability, availability and robustness. A system in which peers work for each other to share the content, with the possibility of centralized content management.

ACM Classification Keywords
C.2.4. Distributed Systems: Distributed applications

Author Keywords
File sharing; Streaming; Music; Content distribution network

General Terms
Documentation, Design Document, Work-in-Progress

Motivation

Multimedia content distribution accounts for up to 50% of global Internet traffic, with the largest provider (Netflix) accounting for almost 40%.
Distributing such massive amounts of data in a traditional client-server setting implicates enormous data-centers across the globe.
Utilizing a peer-2-peer content distribution network; could eliminate the need for these data-centers.

DESIGN AND IMPLEMENTATION

USER INTERFACE

Our optimal user interface would be one alike modern music distribution services, such as Spotify, however as we'd rather focus on the technical aspects, we'll just be implementing a simple HTML interface; featuring a search field, a simple music player and debugging information galore.

TECHNICAL DETAILS

Our content distribution network will be based on BitTorrent for data sharing, and some yet undecided system for searching, replication and content management.

We want to run our system entirely within the context of a modern web-browser, i.e. without being dependent on a backend server. To achieve this, we'll utilize WebTorrent, which features a WebRTC implementation of the BitTorrent protocol.

The system will utilize a central bootstrapping component and/or tracker. Peers will upon joining the system start seeding previously downloaded content, and / or download (and eventually seed) unhealthy content (to ensure availability).

Popular content will be held by more peers than unpopular content, by the fact that peers will seed previously downloaded content. Once a peer reaches a specified resource limit, it will evict content based on some metric; ensuring both high availability and high performance of the system.

We envision that we'll be utilizing some base peer-2-peer protocol for our searching, replication and content management; and simply implement our rules on top.
- We're currently considering a Kademlia based DHT system for this task, but we're wary of implementing anything, but exact-match searching in such a system.

TECHNOLOGIES

Below is a list of technologies we envision utilizing;

- WebTorrent
  - WebRTC
  - Browserify (Web browser nodes)
  - Node.js (Desktop nodes)
- HTML5
  - <audio> tag
  - Local Storage

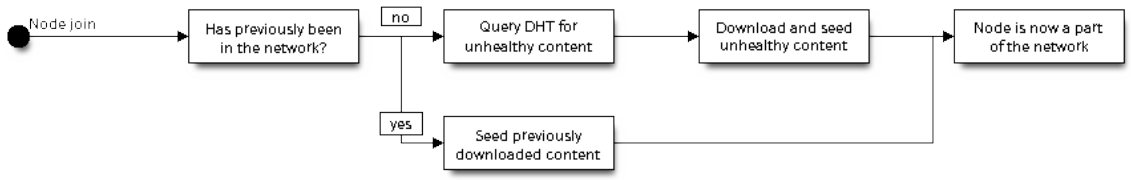
Node join
Has previously been in the network?
no
yes
Query DHT for unhealthy content
Download and seed unhealthy content
Node is now a part of the network
Seed previously downloaded content

Figure 1. Node joining

- Kademlia (DHT)
  - ... or any other WebRTC based DHT

MILESTONES

Our milestones are structures such that we iteratively improve our system.

By week 16 our system should be ready for a rudimentary demonstration, i.e. playing back music acquired via. BitTorrent within the browser.

By week 18 our system should be equipped to search for music in the network.

By week 19 our system should handle replication, and week 20 is mostly 'nice-to-have's, which may be converted into debugging time, or finishing off previous weeks if required.

Week 15:
- Distributing music files between browsers using WebTorrent, via. form-field provided magnet links.
  - Initial seed may be an ordinary torrent client.

Week 16:
- Create magnet links and start seeding, by uploading music files directly in the browser.
  - Magnet links will be distributed via. our content management system.
- Play acquired music files in the browser, i.e. implement a tiny media player.

Week 17:
- Initial investigations of content management, i.e. which systems to use for it.
- Testing of systems, eventually picking a system to use.

Week 18:
- Implement magnet link sharing on the content management system.
- Exact-match searching; i.e. provide file-name and get magnet link for it.

Week 19:
- Implement availability boosting / seeding of unhealthy content.
- Implement seeding eviction, i.e. when a peer reaches it's resource limit.

Week 20:
- Implement a streaming approach when acquiring files (as opposed to ordinary torrent downloading).
- Implement fancy searching, i.e. sub-string, relatedness, artist / genre based, ect.

Future work:
Below is a list of 'nice-to-have's that we'll probably never get around to implement.

- Centralized content management; possibly already handled by utilizing private tracker and limiting access to peers, when authorization is rejected.
- Pretty user interface; to ease usability and acceptance.
- Adapters for external content providers; i.e. gateways to YouTube content, etc.
- Intelligent eviction strategy; formally proven to be optimal for availability and performance.
- Trackerless; Move to a trackerless BitTorrent architecture, i.e. only rely on central bootstrapping on first launch / cleared browser storage.
- Legal training; to avoid lawsuits coming our way, when the system gets adopted by ill-doers for illegal music distribution.